Agile-Lean Software Engineering (ALSE)

Evaluating Kanban in Systems Engineering

A013 - Final Technical Report SERC-2013-TR-022-2

March 6, 2013

Principal Investigator: Richard Turner, Stevens Institute

Co-Principal Investigator: Ray Madachy, USC

Team Members

Jo Ann Lane, USC

Dan Ingold, PhD Candidate, USC

Laurence Levine, PhD Candidate, Stevens Institute

WHS TO 020, RT 035a

Report No. SERC-2013-TR-022-2
March 6, 2013

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **06 MAR 2013**	2. REPORT TYPE	3. DATES COVERED **00-00-2013 to 00-00-2013**

4. TITLE AND SUBTITLE **Agile-Lean Software Engineering (ALSE) Evaluating Kanban in Systems Engineering**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Systems Engineering Research Center,Stevens Institute of Technology ,1 Castle Point on Hudson,Hoboken,NJ,07030**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This research project evaluates the use of on-demand (pull or kanban) scheduling approaches in systems engineering (SE). Such approaches have been seen to be valuable in software system development. In particular, the research focuses on SE where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. Phase I considered the problem and possible applications of alternative scheduling methods and suggested possible outcomes of on-demand scheduling coupled with a service-oriented approach to SE. It defined a conceptual model and developed initial simulations to capture the model and better understand the impact. Phase II focuses on applying the method to multi-level service-based SE in complex systems of systems. Using the models and simulations from Phase I, Phase II defines a prototype network of kanban-based scheduling systems (KSS) for a target environment based on a large multi-facility hospital system. The definition is simulated to demonstrate its behavior. Follow on work is planned to use the prototype in comparing performance with traditional SE methods. This will enable determination if SE functions are accomplished more effectively and efficiently, whether the overall value of the systems of systems over time is increased, and whether other expected results are fulfilled.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**82**	

This material is based upon work supported, in whole or in part, by the U.S. Department of Defense through the Systems Engineering Research Center (SERC). SERC is a federally funded University Affiliated Research Center managed by Stevens Institute of Technology

ABSTRACT

This research project evaluates the use of on-demand (pull or kanban) scheduling approaches in systems engineering (SE). Such approaches have been seen to be valuable in software system development. In particular, the research focuses on SE where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems.

Phase I considered the problem and possible applications of alternative scheduling methods and suggested possible outcomes of on-demand scheduling coupled with a service-oriented approach to SE. It defined a conceptual model and developed initial simulations to capture the model and better understand the impact.

Phase II focuses on applying the method to multi-level service-based SE in complex systems of systems. Using the models and simulations from Phase I, Phase II defines a prototype network of kanban-based scheduling systems (KSS) for a target environment based on a large multi-facility hospital system. The definition is simulated to demonstrate its behavior.

Follow on work is planned to use the prototype in comparing performance with traditional SE methods. This will enable determination if SE functions are accomplished more effectively and efficiently, whether the overall value of the systems of systems over time is increased, and whether other expected results are fulfilled.

ACKNOWLEGEMENTS

This work could not have been accomplished without the support of an extremely talented volunteer industry working group of experienced professionals with extensive experience in lean, kanban and systems engineering. This group included:

- David Anderson (David J. Anderson and Associates)
- Jabe Bloom (The Library Corporation)
- Mike Burrows (David J. Anderson and Associates)
- Ian Carroll (ThoughtWorks, UK)
- Brian Gallagher (Northrop Grumman)
- Hillel Glazer (Entinex)
- Curtis Hibbs (Boeing)
- Suzette Johnson (Northrop Grumman)
- Dominic Lepore (Stevens Institute, Howe School)
- Larry Maccherone (Rally Software)
- Don Reinertsen (Reinertsen & Associates)
- David Rico (Boeing)
- Garry Roedler (Lockheed Martin)
- Karl Scotland (Rally Software, UK)
- Alan Shalloway (NetObjectives)
- Neil Shirk (Lockheed Martin)
- Neil Siegel (Northrop Grumman)
- James Sutton (Jubata Group)

Thanks are also due to the members of the SERC Research Council, particularly Barry Boehm and Jon Wade, for their support.

TABLE OF CONTENTS

FIGURES AND TABLES

This page intentionally left blank

1 SUMMARY

1.1 PURPOSE OF RESEARCH

This research evaluates the use of on-demand (pull or kanban) scheduling approaches in systems engineering where rapid response software development projects incrementally evolve capabilities of existing systems and/or systems of systems. It is hypothesized that such pull systems could provide more effective integration and use of scarce systems engineering resources, enhance flexibility and predictability over complex master schedules, improve visibility and coordination across multiple projects, lower governance overhead, and achieve higher system-wide value earlier.

1.2 WORK ACCOMPLISHED

Phase II focused on applying the method to multi-level service-based SE in complex systems of systems. Using the models and simulations from Phase I, Phase II defines a prototype network of kanban-based scheduling systems (KSS) for a target environment based on a large multi-facility hospital system. The definition is simulated to demonstrate its behavior.

1.3 FINDINGS

The research has shown it is possible to construct a prototype for a hypothetical system using the concepts identified in the Phase One work. We have gained great insight into the complexities of the environment and the flexibility required for such a system to be considered, implemented and deployed.

The concepts of a pull system, reduced work in progress, and systems engineering as a service have all been included in the hypothetical system. However, much of the benefit in the system is from the interactions of the various stakeholders and engineers and the conversations that are held when applying the techniques. These are not the type of information exchanges that are commonly seen in such environments. Simulation runs, charts, and tables simply do not capture this aspect of the approach.

The work is now sufficiently well understood to attempt to pilot some or all of the concepts in vivo, although it will not be a simple task. Discussions are under way with two different industry organizations to take that step.

As with all research, you may not be able to accomplish what you set out do. In this case, we fell short in creating useful simulations that handled the nondeterministic character of the environment. We still have a lot of work to do in order to productize and support application of the concepts.

1.4 Research Results

Beyond the findings as presented, the following research accomplishments have been achieved:

- An international advisory working group has been established and has contributed to this work.
- Five peer-reviewed conference papers have been accepted
- Two non-paper conference presentations have been given
- One invited paper has been accepted
- Two international conference workshops on the subject have been conducted
- Three doctoral candidates (Dan Ingold, USC; Laurence Levine and Robert Carlson, Stevens) are using this work in their research

1.5 Next Steps

Follow on work is planned to use the prototype in comparing performance with traditional SE methods. This will enable determination if SE functions are accomplished more effectively and efficiently, whether the overall value of the systems of systems over time is increased, and whether other expected results are fulfilled. Pilot projects are also planned to validate the approach in vivo.

2 INTRODUCTION

Traditional systems engineering (SE) developed over half a century ago, primarily driven by the challenges faced in the aerospace and defense industries. The environment was fairly uniform – hardware-driven, long lived, single mission. The result of this uniformity was practices that worked well in that specific context were seen as "best practices," and came to define the discipline of systems engineering. In the last few decades, system contexts have multiplied, and the speed of change in both needs and solution technologies has accelerated. This has led to an inherent loss of determinism—requirements are less tangible, more evolving, and sometimes emergent and systems are both complex and constantly adapting. The practice of systems engineering, with its roots in long-term, primarily hardware projects, has not kept pace.

Engineering principles involving agility and leanness have been adopted to address non-determinism in software systems. They use iterative and spiral concepts, require less traditional ceremony, maintain closer interaction with stakeholders, and are based on best practice, underlying theory and overarching principles. Combining agile-lean software experience with system engineering fundamentals can provide practical, principle-driven agile-lean systems engineering approaches for the design of complex or evolving hardware-software-human systems.

This ongoing research examines one of those approaches, kanban (pull) scheduling techniques, to determine its applicability to systems and software engineering in a rapid response environment. Derived from lean manufacturing and modified significantly to apply to knowledge work such as software engineering, pull scheduling aims to provide the highest value flow through a system by visualizing the work in progress (WIP) and only starting additional work when some resource is available to do it – usually when another piece of work has been completed. It applies WIP limits, selection based on next highest value, and Classes of Service to manage flow. For those not familiar with this concept, Appendix B provides an introduction based on previously published work. [24, 25, 26, 27]

The rationale for attempting to apply this concept to systems engineering is based on expected benefits that have accrued to organizations that have used it in software and in services organizations. The benefits are:

- More effective integration and use of scarce systems engineering resources
- Enhanced flexibility and predictability
- Better visibility and coordination across multiple projects
- Lower governance overhead
- Increased project and system value delivered earlier

These benefits are described more fully in Appendix B.

2.1 Research Goals and Approach

While our initial efforts confirmed our belief that KSS and service-oriented SE can be effective, the research sponsor tasked us to investigate how the KSS concept could be applied to the problems of multi-level SE (MLSE) or systems of systems engineering (SoSE). These are cases where system engineering is not focused on a single system, but on a set of integrated but often disjoint systems that share common services but may have totally different management or oversight mechanisms. The amount of information required to visualize the work in progress, the interdependency among many often concurrent and sometimes intentionally compartmented activities, the need to handle multiple types of tasks across several layers of abstraction, and the multiple value systems that operate across multiple systems make the concept of managing flow much more complex than a simple software product development team.

The model and simulations developed in Phase I were used as a basis for adapting the KSS concept to MLSE, but more specific details of the information needed to accomplish such a complex KSS were required. We had theorized that KSSs could be networked, but had not actually modeled the concept to any useful level of detail. We determined that the generic description we had was sufficient, but that the most reasonable approach was to find a target environment that reflected the issues the sponsor was concerned with and then approaching it as if we were a consulting team trying to solve a specific problem using a KSS Network.

Jo Ann Lane, one of our researchers had worked on a complex information system for a large, multi-sited hospital system that had many of the same characteristics as our sought after target environment. The hospital system supported several hundred geographically dispersed facilities. Because of her knowledge and the criticality of hospital systems, this seemed a particularly good target environment. We describe the simplified version of this system in Chapter 3.

Based on her understanding of how the systems and software engineering work was managed within the systems development organization, we developed a set of goals for the new system. The new approach uses a network of integrated KSSs that are designed to

1. Make visible work in progress
2. Establish and track organizational capacities at all levels
3. Limit WIP to improve value flow (identify resource issues, cause of blocked work)
4. Coordinate multiple levels of systems engineering activity
5. Communicate progress with respect to senior management goals
6. Support analysis and decision making at every level of management
7. Make visible current progress toward development and deployment of capabilities
8. Establish a basis for continuous improvement in a rapidly changing environment

The KSS Network shows the relationships between the software development tasks and the systems engineering tasks. It also clearly captures the relationships between both the software and systems engineering tasks and the required capabilities.

Because kanban concepts have been primarily used with single level value streams, we wanted to understand the information needed for decision making, including scheduling and flow monitoring/control, at each level of SE activity or utilization. This would allow us to construct a KSS that would support visualization of WIP and status for each specific level. It would also provide insight into the information flow required.

To accomplish this, we turned to the Basili, Caldieri, and Rombach Goal Question Metric approach [12,13]. For each level we defined the goals and the questions that made sense to ask in order to determine if the goals were being met. Given our research is to investigate KSSs, we decided to fully utilize the flow and pull concepts. So, rather than identify metrics to answer the questions, we determined the information that each KSS would provide for that purpose, referring to the approach as goal-question-kanban (G-Q-K). This is an ongoing process, and we will continue evolving the G-Q-K within the team and through the working group.

3 Multi-level and System of Systems SE

Multi-level and System of Systems SE can best be described in terms of the seven high-level SoS processes described in [22] and illustrated in Figure 1.

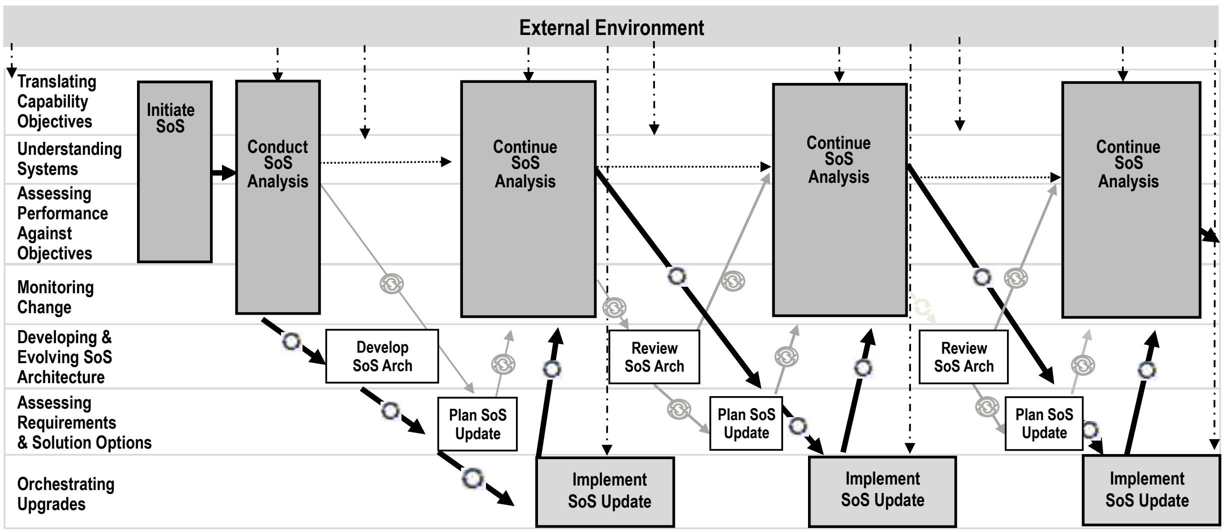


Figure 1. SoS engineering process (Adapted from [23])

Briefly, these high-level SoS processes can be described as follows:

- **Translating capabilities to requirements**: The SoSE team must develop a basic understanding of the expectations of the SoS capability and then translate the capability into a set of requirements for meeting the expectations.
- **Understanding systems (products) and relationships:** In a SoS, the focus is on the systems (products) that contribute to SoS capabilities and their interrelationships instead of the traditional focus on boundaries and interfaces.
- **Assessing SoS performance:** To be able to understand current SoS performance and ascertain the impact of constituent-system (product) changes, the SoSE team establishes SoS metrics, defines methods for assessing performance, and conducts evaluations of actual performance using the metrics and methods.
- **Developing, evolving, and maintaining an SoS architecture/design:** As soon as systems (products) start interfacing with each other and sharing data, there is an implied architecture for the collection of systems (or SoS). One of the key responsibilities of an SoSE team is to establish and maintain a sustainable framework to support the evolution of the SoS to meet user needs. Evolutionary changes include changes in systems' (products') functionality, performance, or

interfaces. These needed changes often require systems (products) to migrate from the early "implied" architecture to a more robust architecture or framework.

- **Monitoring and assessing changes:** The SoSE team must constantly monitor proposed or potential changes to the constituent-systems (products) and assess their impacts to a) identify opportunities for enhanced functionality and performance, and b) preclude or mitigate problems for the SoS and other constituent-systems.
- **Addressing new requirements and options:** The SoSE team reviews, prioritizes, and determines which SoS requirements to implement next. Part of this activity is evaluating various options for implementing the capability and requires the participation of the affected constituent-systems (products).
- **Orchestrating upgrades to SoS:** This activity is the actual implementation of the desired capabilities and includes the planning, coordination, integration, and testing of changes in the constituent-systems (products) to meet SoS needs. Note that the SoSE team does not implement the actual changes, but flows the requirements to the constituent system (product) teams for implementation, then engages again during the integration, test, and deployment activities.

The research described in this technical report focuses on all of these activities at the SoS level as well as the management/stakeholder activities above the SoS level and the product (or constituent system) systems engineering and software development activities below the SoS level.

4 HEALTH CARE SYSTEM OF SYSTEMS DEVELOPMENT ORGANIZATION AS-IS DESCRIPTION

4.1 OVERVIEW

The health care SoS is a medical information management set of integrated systems that consists of hardware, several million lines of source code, numerous commercial-off-the shelf (COTS) software products, and communications networks that support the administration and delivery of health care in networked set of hospitals and clinics.

4.1.1 Health Care Development Organization

The Health Care Development Organization consists of three groups, as shown in Figure 2.

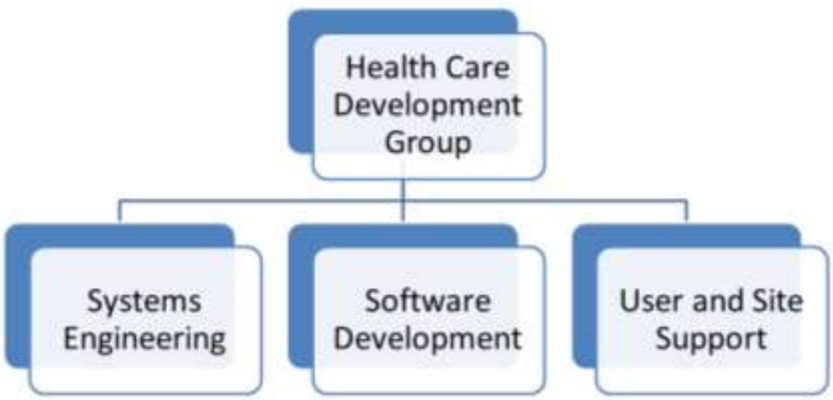


Figure 2. Health Care Organization

The systems engineering group is responsible for make-versus-buy trade studies related to new capabilities or enhancements that might be provided by COTS products, conducting evaluations of candidate medical devices for integration into the health care SoS, system performance assessments of both deployed systems and system enhancements under development, networks for both the deployed systems and the development environments, hardware and network upgrade recommendations, security engineering, constituent system integration, and system and SoS-level acceptance testing.

The software development group is responsible for software maintenance and enhancement for the custom Health Care constituent systems or products; database structures and embedded procedures; COTS product tailoring, glue code, integration, and upgrades; and licensed data upgrades such as the pharmacy approved formularies, as well as responding to software issues that are beyond the scope of the user help desk.

The user and site support group is responsible for running the user help desk, site configuration management, and site installations and upgrades.

4.1.2 Scope of Health Care System Capabilities

The key custom software constituent systems within the health care SoS include user access management, patient management, pharmacy, laboratory, radiology, and patient telemetry. The constituent systems share a single database that maintains the information for all of the patients and personnel related to a given health care site. Some key constituent systems are supported by tailored COTS products and integrated into the health care system. In addition, there are interfaces to other health care systems maintained by the parent organization at various sites. The interfacing systems include custom legacy systems, COTS products, and electronic medical devices such as heart rate monitors and infusion pumps.

4.1.3 Health Care System Engineering Professionals

Currently, there are over 1,000 engineering professionals working on this system, of which about one third work in software development. The other two-thirds work in system engineering, system integration and test, or customer service and support. At any given point in time, work is in progress on several releases of the software. A release may be a formal major release or it may be a minor maintenance update.

4.1.4 Health Care System Key Goals and Challenges

The health care system's primary goal is to support patient health care and to provide health care in a timely and safe manner that is coordinated across a variety of health care providers and specialists. Key overarching requirements are to ensure patient-safety and to protect patient information in accordance with government Health Insurance Portability and Accountability Act (HIPAA) and other privacy and security regulations. To meet these goals and regulations, the health care organization provides periodic software and system updates.

Currently, the software development group has reported problems with getting software upgrades to users because of a lack of needed systems engineering support. Software upgrades are blocked waiting for systems engineering activities such trade studies, security assessments, hardware and network performance assessments, and hardware and network upgrades.

4.2 Current Engineering Organization Structure and Process

The current systems engineering and software engineering organizations are fully staffed with respect to the annual health care SoS development and maintenance budget. Figure 3 illustrates at a high level how new needs (or capabilities) are currently handled by systems and software engineering. When new needs or capabilities are identified, systems engineering analyzes the new needs/capabilities in terms of the given systems and decides how address them. Often multiple new needs/capabilities are analyzed together to facilitate the identification of common solutions that can

support more than one need/capability as well as support performance upgrades and technology refresh.

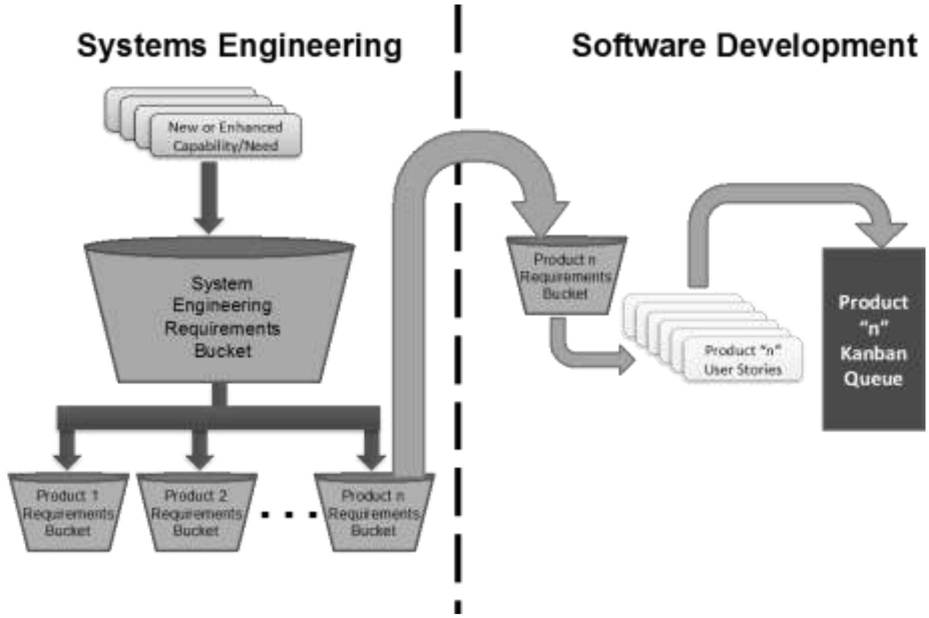


Figure 3. Flow of tasks from SE to Software Product Queues

The results of the analysis activities are a set of requirements. The next step in the process is to allocate those requirements to one or more products for implementation. Figure 4 provides an example that illustrates how multiple requirements are derived from one or more needs and then mapped to the enterprise products for implementation. Figure 4 also shows in more detail how well-engineered capabilities can use common solutions and requirements can often map into more than one constituent system/product.

Finally note that at this level, the requirements tend to be distributed between three high level categories. Some of the requirements are related to performance enhancements, some to computational/information processing within the products, and others to interfaces between constituent systems that enable the exchange of data/information between the constituents. This highlights the fact that often multiple engineering specialists are needed to support upgrades for software-intensive systems and SoS. This range of specialists can include security engineers; safety engineers; hardware and network specialists to ensure acceptable transaction throughput, user response time, and system availability; as well as domain experts that in the healthcare area would include physicians, pharmacists, and radiologists.

Once the requirements are allocated to the products, the product teams analyze them and convert them into user stories for implementation. Implementation is an incremental process composed of 90-day spins. User stories in the product backlog are evaluated and prioritized and the highest priority user stories are assigned to spins. Figure 5 illustrates a logical data model showing how capabilities are decomposed to the point where the associated requirements are allocated to increments and spins.

Note that in most SoS, all product constituents are not upgraded at the same time. Rather, each constituent system (or product in the case of the healthcare example) has their own increment deployment schedule. Sometimes product schedules coincide with each other, but often not.

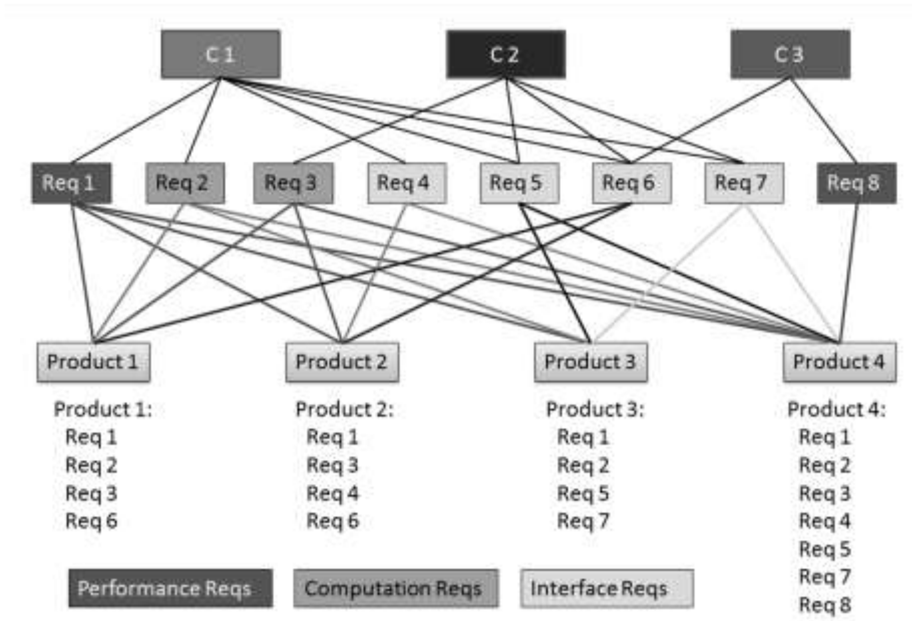


Figure 4. Capabilities to requirements to products

Once an implementation strategy is defined for a deployable increment, systems engineering needs to conduct operational performance assessments for all of the sites that will receive the new increment. This assessment looks at the additional load the new increment will place on the operational environment, in particular with respect to existing hardware and network resources. For some planned changes, security/patient safety assessments may be required and may cause changes to security policies and procedures as well as the need for product re-certification before the increment is deployed.

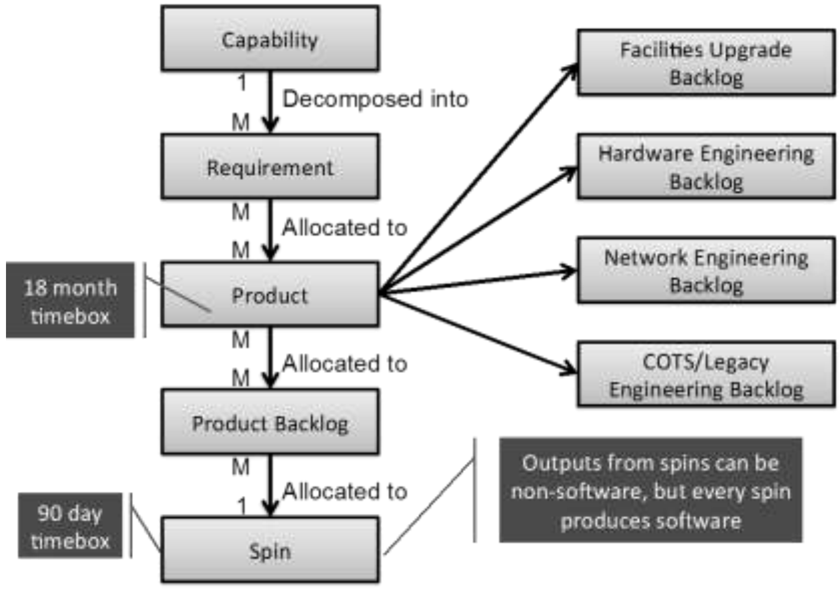


Figure 5. Mission capability decomposition and allocation.

The software development team also works to identify opportunities to incorporate lower priority changes with the higher priority changes, especially when the area changed may require recertification or expensive testing. This approach minimizes the amount of testing and recertification over time and provides opportunities to implement the lower priority changes that by themselves do not justify the time and expense of recertification.

Finally, systems engineering monitors the capability "pieces" to guide their system integration, testing activities. When all of the capability requirements are implemented in the affected products and deployed, the mission capability is considered "completed".

However, the software engineering organization reports that many of their tasks become blocked waiting for systems engineering tasks to complete. As a result, many tasks are started, but difficult to complete in a timely manner.

In addition, there is no visibility at the capability level, showing which user stories are related to which capabilities and which products are implementing pieces of the capability.

5 Designing the Prototype KSS Network

5.1 Overview

To improve the software development flow and to enhance senior management visibility into the development process, a new three-tiered Kanban process has been proposed. Through analysis of the organizational information needs, a set of Kanban levels and a set of information have been defined for the each Kanban level.

The proposed Kanban levels are:

- Executive/Stakeholder Management (ESM)
- Capability Engineering (CE)
- Product/Domain Engineering (PDE)

Figure 6 provides an overview of the Health Care System KSS-network. Each shows function levels, KSS elements for monitoring and control, and Dashboards for providing information from multiple KSSs.

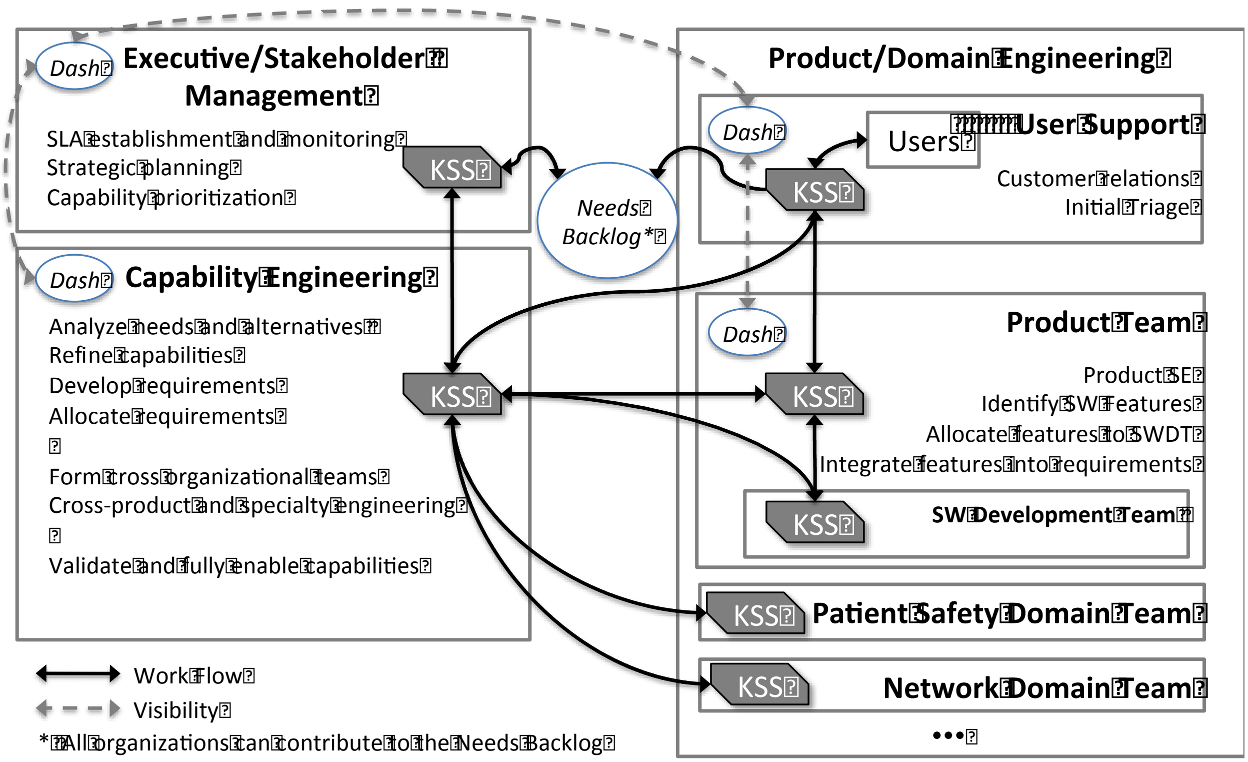


Figure 6. Overview of KSS Network

5.1.1 Executive/Stakeholder Management Level

The ESM level is the level that determines which proposed capabilities (or capability enhancements) are going to be approved for development. As part of this process, the ESM level assesses the value of the capability against its expected cost and schedule to develop. In addition, this highest-level in the KSS network is concerned primarily with the current status of identified capabilities (or needs) as represented by the development state of each "not fully deployed" but "approved for development" capability – essentially WIP. At this level, the KSS is tracking capabilities and their priority. The insight it provides should inform decisions about overall organizational strategy, resource staffing, and development funding priorities. Table 1 presents an example G-Q-K strategy at this level.

Table 1. Example Executive/Stakeholder Management Goal-Question-Kanban Framework

Goals	Questions	KSS Information
G1. Deploy capabilities according to value-based priorities and CoS. G2. Understand source/cause of blocked work flows G3. Strategic IT decisions based on current and projected WIPs and backlogs (examples might include investments in additional resources (hardware, tools, people) or decisions to drop lower priority capabilities). G4. Changing needs and priorities are integrated with existing strategy	Q1. What capabilities are currently in progress? Q2: What capabilities are currently blocked? Q3: What capabilities are pending acceptance? Q4. What is the planned and actual organizational value of each deployed increment? Q5: What is the balance between backlog and WIP? Q6. What is the average time to completion for "accepted" capabilities (by CoS)? [If the difference in capability effort is significant, this could be the average difference between predicted and actual lead times) Q7. What is the requirements change rate by capability? Q8. What KSSs show capacity not meeting demand? Q9: What KSSs indicate excess capacity? Q10. What are the current WIP levels at the Capability, SE, and Product levels?	KSS1: Flow data on SoSE and Product Teams* KSS2: Average time to deploy capabilities for each CoS priority level KSS3: Relationships between capabilities and requirements KSS4: Status of requirement completion/deployment KSS5: Percentage of requirements completed/deployed for each in-process capability KSS6: Status of SE tasks supporting capability acceptance decisions **Includes Cumulative Flow Diagrams (CFD) (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data.*

5.1.2 Capability Engineering

This level represents all capability-related SE activities, specialty SE support for product teams, including software system engineering tasks, where software is a key component in the requirements allocation. CE is responsible for creating capability descriptions that incorporate the needs identified and prioritized by the ESM level.

CE must balance the various SE resources as they work with both internal activities and lead cross-organizational teams in CE-related activities. Decisions and scheduling of the SE resources must include front-end and ongoing architectural work as well as the day-to-day support for development, integration, verification and validation that interacts directly with the various product teams. Table 2 presents an example G-Q-K strategy at this level.

Table 2. Example Capability Engineering Goal-Question-Kanban Framework

Goals	Questions	KSS Information
G1. Cost-effective and timely alternatives identified for new capabilities/capability enhancements G2. Adaptable, flexible, multi-purpose solutions provided for new capabilities/capability enhancements G3. Specialty engineering responses to software teams' SE requests do not create excessive delays in capability development G4. Provide quick response to changing needs and priorities	Q1. What work is currently blocked? Q2. What is the % of capabilities that are deployed within the desired timeframe? Q3. What is the predicted time to completion for "accepted" CE tasks (by class of service)? Q4. Where is capacity not meeting demand (by capability specialty engineering discipline)? Q5: Where is there excess capacity (by capability specialty engineering discipline)? Q6: What is the age of items in the CE backlog queues? Q7. What are the current CE WIP levels? Q8. What are the current CE backlog levels? Q9. What is the balance between CE WIP and CE backlog?	KSS1: Flow data on SoSE and Product Teams* KSS2: Average time to deploy capabilities for each CoS priority level KSS3: Relationships between capabilities and requirements KSS4: Status of requirement completion/deployment KSS5: Percentage of requirements completed/deployed for each in-process capability KSS6: Status of SE tasks supporting capability acceptance decisions **Includes CFD (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data*.

5.1.3 Product/Domain Engineering

At the PDE level, there are separate KSSs for each product or domain team in the enterprise. The KSSs at this level are similar to those in use in many software development organizations today, with the added requirement to perform systems engineering within the product or domain scope. What is different for those constituent systems/products that participate in one or more SoSs, is the need to provide information to higher level KSSs and dashboards all the way up to the ESM level.

The User Support Team operates at the PDE level because it primarily interfaces with the product and domain teams. There are occasions, however, when it influences the needs backlogs, or when it uncovers an issue (e.g. patient safety or

privacy) that requires engagement with ESM and CE to handle the solution. Each product or domain team is responsible for implementing capability-related requirements allocated to that product as well as responding to User Support problems that cannot be handled by the User Support team. These User Support tasks are typically fixes to reported problems of varying severity levels, some of which require immediate attention; in the health care example, these might be issues that affect patient safety or privacy.

In addition, each product team may have a somewhat unique team organization depending on whether it is internal or outsourced. If outsourced, contractual requirements, corporate size and corporate governance will influence the KSS implementation. For example, if the outsourced organization operating the product team uses a matrix organization for SE, there may be a separate KSS defined for the SE resources that may cross product team boundaries. If the contractual SE resources are each dedicated to a specific product, then their tasks can be included in the individual product team KSS or the software development team KSS.

Table 3 presents an example G-Q-K strategy at the PDE level and assumes that it includes the Product-Level SE tasks.

Table 3. Example Product Goal-Question-Kanban Framework

Goals	Questions	KSS Information
G1. Capability-allocated requirements are developed and deployed according to their product value G2: Product requirements and features are allocated to increments and spins based on value G3. Product team responds quickly to changing product needs and priorities G4. Minimize workflow disruptions in product increments and spins G5. Minimize rework due to poorly understood capability requirements G6. Product team provides timely responses to user support issues/problems	Q1. Value of product-level work currently blocked? Q2. What is the % of requirements completed within the desired timeframe? Q3. Where is PD capacity not meeting demand? Q4: Where is there excess PD capacity? Q5: How often are the average age of items in product-level backlog queues within expected levels? Q6. What are the current product-level WIP levels? Q7. What are the current product-level backlog levels? Q8. What is the product-level response time to SW requests?	KSS1: Flow data on Product Team* KSS2: Number/status of tasks in product-level queues (initial analysis, backlog, WIP, blocked) KSS3: Number of tasks in product-level queues that are blocking other tasks (e.g., dependent tasks) KSS4: Relationships between capabilities, requirements, and features at product level KSS5: Percentage of each in-process requirement already completed/deployed KSS6: Average User Support request task completion time **Includes CFD (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data.*

5.2 Classes of Service

Classes of service (CoSs) provide a variety of handling options for different types of work and affect the next work item selection value for KSSs. They may be aligned with Service Level Agreement (SLA) priorities. Their implementation allows the WIP limits to be distributed in such a way that certain types of work will always take priority, will have more consistent access to resources, or will only be selected under certain circumstances. Most CoSs are intended to ensure priority rather than force immediate execution. There are CoSs that are disruptive – that is, they can suspend current work in progress. This allows swarming of all appropriate resources to ensure completion as soon as possible. Disruptive CoSs are minimized to the extent possible because they act counter to the normal kanban philosophy of completing work rather than having a large WIP. They are usually associated with critical or expedited work.

5.2.1 General CoSs

For the healthcare case study, we have defined five general classes of service (COS) that apply to all the work in the KSS Network. Different environments and situations would require different, more, or less classes of service.

5.2.1.1 Critical Expedite

A Critical Expedite work item represents something that fixes an existing or imminent issue within the system. Safety, security, or other emergency work items are assigned this CoS. It is disruptive and requires all appropriately skilled resources to suspend their current activities and work on the Critical work item. It also suspends any WIP limits in activities associated with its work items for the duration that the critical work is in the activity. It applies to every allocated work item associated with the work item assigned to this COS in all KSSs regardless of local priorities.

5.2.1.2 Important

The Important CoS is assigned to very high priority work items where the speed of completion is such that this work should take priority over all other work in the ready queue. It is not disruptive, because all WIP is allowed to finish before the important work begins. It does not impact WIP limits, but has a guaranteed WIP limit in some KSSs.

5.2.1.3 Date Certain

Date certain (or schedule as independent variable) class of service reflect work items that must be completed by a specific date or there will be significant consequences. Regulatory implementation deadlines, COTS upgrade preparation, or integration/deployment dependencies are candidates for this class of service. It operates essentially like an Important CoS, but as the date becomes closer, it may elevate to Critical Expedite based on workload.

5.2.1.4 Standard

This is the normal CoS for the development organizations work. A high percentage of work should be assigned at this level for the KSS Network to provide the desired outcomes.

5.2.1.5 Background

Background work (sometimes referred as intrinsic or invisible) is work that must go on but is usually not time critical. It includes things like architectural enhancements, low-level technical debt, research and environmental scanning, or time-certain events not due in the near future. It is usually prioritized by its length of time in the queue (FIFO). Some KSSs may have a limit for the time background work can remain in the queue. When reached, this limit automatically places the work item in another CoS.

5.2.2 KSS-Specific CoSs

Most CoSs are shared across the entire KSS network. However, individual KSSs may define additional CoSs to handle flow specific to their types of work.

5.1 KSS Description

This section defines how we describe the suggested operation of the various KSSs in the KSS Network. Each KSS is based on the workflow, the G-Q-K definitions, and the special circumstances and needs of each organization of resources represented by the KSS.

There are nearly as many ways to define a KSS as there are to define a system. In this case, we recommend processes and visualizations appropriate to our target organization. Many other approaches are possible, and as we continue our research, we will try to identify any patterns or anti-patterns that occur. As we complete the simulations, we will try different strategies and evaluate them against value delivered over time, handling of critical and important work, and relative stability of flow.

Each KSS design has a summary, processes for work flow, and one or more visualization tools.

5.1.1 KSS Template

The template shown in Table 4 is used to summarize each KSS in the network. It identifies key factors about the KSS, including what organizations can assign work to the KSS demand queue, the activities that are tracked by the KSS, flow controls available, and information about the type of resources available to the activities.

Table 4. KSS Template

KSS Name	
Demand:	
Work sources	*Organizations that can assign work items to the KSS*
Resources:	
Dedicated	*Resources under control of this KSS*
Shareable	*Resources available to share on teams with other orgs*
Sourced	*Organizations (KSSs) to which work items can be assigned*

Managed resource specialties	*Any specialists that are managed individually*		
Activities:			
Description	WIP Limit	Resource Type	Cohesion
		Internal, Sourced, or X-discipline team	*Interruptible or Must complete*
Flow and Visibility:			
Additional CoS handled	*CoS beyond the system-wide that are recognized by this KSS*		
Additional CoS introduced	*CoS defined for work this org assigns to other KSSs*		
Work Selection Value Adjustments			
Source-based	*CoS-based*	*Resource-based*	*Completion-based*
Goals	*From GQK*		
Questions answered	*From GQK*		
Data maintained/used	*From GQK*		
Information shared	*e.g. Avg. Lead time, Avg. blocked tasks. Avg. time blocked, Avg. resource WIP, Avg. Backlog length, Statistical limits for information types,*		

5.2.3 KSS Flow Process Descriptions

These descriptions provide insight into the information and activities within the KSS. There are several processes common to each KSS. Other processes may be identified for a specific KSS.

5.2.3.1 Accepting/Selecting Next Work Item

This process describes how work items are selected from the demand backlog. This is a highly personal and collaborative process that is closely related to the resource allocation process. It includes the cadence for selection, the limits of the demand queue and any limits on the backlog. This process also addresses constraints on the capacity and on the demand that would impact the performance. It also balances the value of the work to both the demand source and to the performing KSS.

The value of work may change due to adjustments related to the KSS environment. For example, some sources may have inherent priority over other sources for political or other reasons. Classes of Service may be interpreted differently where there is a higher instance of necessary ongoing maintenance tasks so that critical maintenance does not drop behind due t capability or enhancement projects. Resources may also drive manipulating the value. If, for example, a significant number of resources are delegated to cross-organizational work or are absent for other reasons (e.g. military deployment or illness), there might be a lowering of the value of work that might require their expertise until such time as they return. Finally, there can be general rules for special cases. One case we demonstrate here is a value prejudice toward selecting work that applies to a requirement or capability nearing completion. This supports the idea of completing work and thus systematically reducing WIP.

5.2.3.2 Allocating Resources and Team Development

This process handles resource assignment and the formation of teams where required. Every KSS will have different ways of handling this allocation of work. For example,

there may be specific assigned teams or it may be that the first available resource is assigned or self-selects the next piece of appropriate work. This process interacts with the selection process by considering the existing capacity to complete work in the demand queue.

5.2.3.3 Completion and Disbursement

This process specifies any actions that are required when work items are placed in the final KSS done queue. As an example, this could include work collecting and integrating sub-tasks derived from a work item and separately handled that must all be completed before the initial work item is considered "done."

5.2.3.4 KSS Review

This is the process for walking the visualization board or reviewing the dashboard. It sets the cadence for the review, describes the way status is reviewed and resource/blockage issues identified, and what decisions as to resource allocation, work item selection, and incremental process improvement are considered and made.

5.2.4 Visualization Tools

The two keys to the pull scheduling approach are the visibility into work in progress and the ability to resolve flow issues at the lowest levels. These keys are dependent on the efficacy of the visualization tools. In the complex multi-level systems engineering and development environment, the visualization tools will almost certainly be interconnected electronically. How that interconnection takes place is not addressed specifically. Rather, the visualizations are updated according to a cadence that permits them to operate most effectively. The lowest levels may be physical devices mechanically updated on one cadence and electronically recorded for upstream KSSs on another cadence according to the need. There are two specific types of visualization tools used in the Health Care KSS Network: kanban boards and dashboards.

5.2.4.1 Kanban Boards

Kanban boards (Figure 7) are active management and control tools that provide a common operating picture for all resources and work items associated with a KSS. The boards are organized according to demand, activities, and status, and have work items (Figure 8) as their predominant content. They are interactive and updated rapidly to act as both information radiators and operational tools where information is added, consulted, adjusted, and removed as the work flows through the systems.

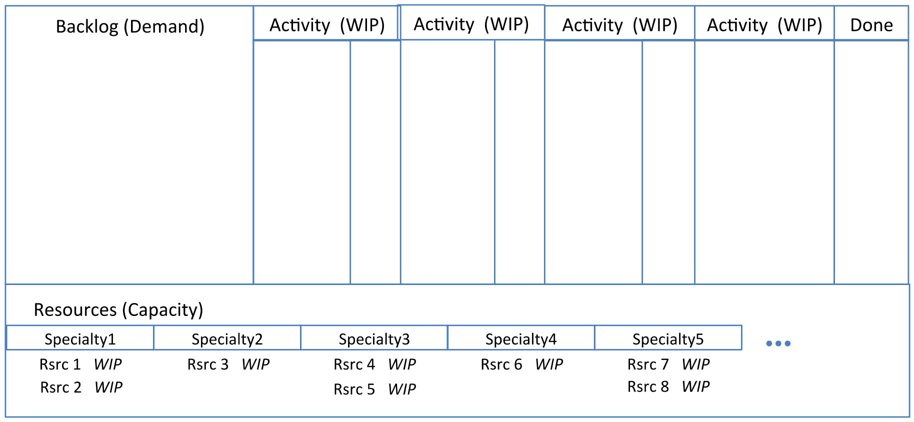


Figure 7. Generic Kanban Board Template

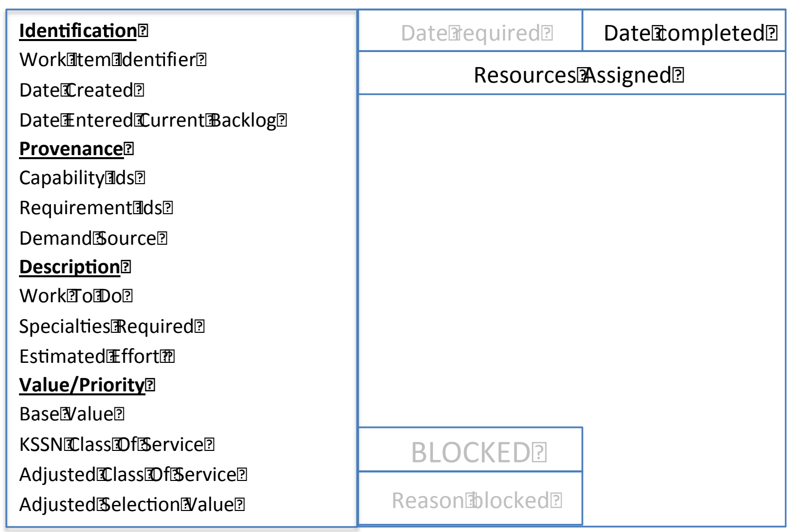


Figure 8. Generic Work Item Description

5.2.4.2 Dashboards

Where multiple KSSs need to be involved in resource management and other decision-making, we define a dashboard (Figure 9) to visualize information gathered by the KSSs, in accordance with the information required by the G-Q-K analysis. Dashboards are pure information radiators designed to quickly communicate specific data useful in status assessment and decision making for the specific area or organizational component. They are usually not interactive and often feature automatic updating, data in context charts (such as graphs or percentages), and scrolling information.

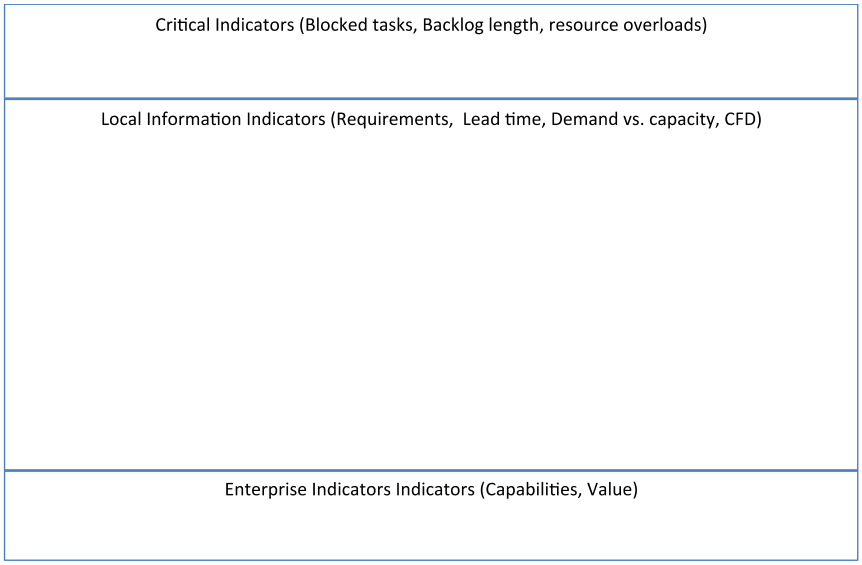


Figure 9. Generic Dashboard Template

6 THE HEALTHCARE KSS NETWORK

This section defines the KSSs in the Healthcare KSS Network in terms of the components described in the previous section.

6.1 EXECUTIVE/STAKEHOLDER MANAGEMENT KSS

The ESM KSS is primarily interested in the performance of the various domains in achieving high value output as quickly as reasonable and in accordance with the established goals. The overall performance of the development organization is illustrated continuously by the work tracked on the kanban board and the summarized information provided by the Dashboard.

Table 5. ESM KSS Template

<table>
<tr><th colspan="5">Executive/Stakeholder Management KSS</th></tr>
<tr><td colspan="5">Demand:</td></tr>
<tr><td>Work sources</td><td colspan="4">Needs backlog, Stakeholders, Critical Events, Strategic Plans</td></tr>
<tr><td colspan="5">Resources:</td></tr>
<tr><td>Dedicated</td><td colspan="4">IT Managers, CTO, …</td></tr>
<tr><td>Shareable</td><td colspan="4">None</td></tr>
<tr><td>Sourced</td><td colspan="4">CE</td></tr>
<tr><td>Managed resource specialties</td><td colspan="4">None</td></tr>
<tr><td colspan="5">Activities:</td></tr>
<tr><td colspan="2">Description</td><td>WIP Limit</td><td>Resource Type</td><td>Cohesion</td></tr>
<tr><td colspan="2">Capability Analysis</td><td></td><td>Sourced (CE)</td><td>Interruptible</td></tr>
<tr><td colspan="2">Capability Prioritization-CoS Assignment</td><td></td><td>Internal</td><td>Must complete</td></tr>
<tr><td colspan="2">Capability Development Project</td><td></td><td>Sourced (CE)</td><td>Interruptible</td></tr>
<tr><td colspan="2"></td><td></td><td></td><td></td></tr>
<tr><td colspan="5">Flow and Visibility:</td></tr>
<tr><td>Additional CoS handled</td><td colspan="4">None</td></tr>
<tr><td>Additional CoS introduced</td><td colspan="4">None</td></tr>
<tr><td colspan="5">Work Selection Value Adjustments</td></tr>
<tr><td>Source-based</td><td colspan="2">CoS-based</td><td>Resource-based</td><td>Completion-based</td></tr>
<tr><td>None</td><td colspan="2">None</td><td>None</td><td>None</td></tr>
<tr><td>Goals</td><td colspan="4">G1. Deploy capabilities according to value-based priorities and CoS.
G2. Understand source/cause of blocked work flows
G3. Strategic IT decisions based on current and projected WIPs and backlogs (examples might include investments in additional resources (hardware, tools, people) or decisions to drop lower priority capabilities).
G4. Changing needs and priorities are integrated with existing strategy</td></tr>
<tr><td>Questions answered</td><td colspan="4">Q1. What capabilities are currently in progress?
Q2: What capabilities are currently blocked?
Q3: What capabilities are pending acceptance?
Q4. Are the planned and actual values of each deployed capability tracking?
Q5: Are the current WIP level for ESM activities correct?
Q6. What is the average time to completion for “accepted” capabilities by CoS?
Q7. What is the requirements volatility by capability?
Q8. What KSSs show capacity not meeting demand?
Q9: What KSSs indicate excess capacity?</td></tr>
<tr><td>Data maintained/used</td><td colspan="4">KSS1: Flow data on CE and Product Teams*</td></tr>
</table>

	KSS2: Average time to deploy capabilities for each CoS priority level KSS3: Relationships between capabilities and requirements KSS4: Status of requirement completion/deployment KSS5: Percentage of requirements completed/deployed for each in-process capability KSS6: Status of SE tasks supporting capability acceptance decisions *Includes CFD (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data.
Information shared	Capabilities under development, CFDs for each Capability, Network Value Tracking,

6.1.1 Executive/Stakeholder Management KSS Processes

At the ESM level, the focus is on capabilities, resource utilization, overall system flow, and strategic issues.

6.1.1.1 Accepting/Selecting Next Work Item

Requests for system capabilities come from the users, from the systems engineering groups, and from strategic initiatives. There is always a backlog of ideas needs, and wants. ESM must identify the highest priority capabilities. They must balance adding new capabilities with improving existing system capabilities and maintaining the infrastructure. They must also act on critical issues regarding patient safety, infrastructure failure, and regulatory changes. The outcome of this process is sending only the highest value and most critical work to the systems engineering group to analyze and develop.

Some work items initiated within the ESM level are special studies related to the prioritization of capabilities and the possible combination of multiple needs into a more effective capability need. This work includes cost and schedule estimations, Ops Concept development, COTS evaluations, and other traditional front end SE activities. Once assigned, these are included in the work flow KSSs at the SoSE and Engineering Specialty organizations.

6.1.1.2 CoS Implementation

ESM assigns CoS to the work it allocates to CE. Within the selection decision process, the five general CoSs allow the ESM leadership to influence the priorities of work associated with capabilities throughout the KSS Network.

6.1.1.3 Allocating Resources and Team Development

ESM must understand the overall capacity, work in progress, and resource distribution across CE and PDE teams in order to determine the highest priority capabilities and decide how to meet strategic needs and balance ongoing tasks. Starting too many capability developments can lead to less effective execution, while starting too few may jeopardize market share or stakeholder satisfaction. This organization must work closely with the CE organization and the User Support Team to map the landscape reflected in the needs backlog.

6.1.1.4 Completion and Disbursement

While the decision to deploy is often a systems engineering or PDE Team decision, the declaration of a capability being "finished" (i.e. fully implemented and deployed) is usually reserved for the ESM.

6.1.1.5 KSS Review

At this level, review of the work in progress, demand, capacity, and performance is focused on achieving capabilities and handling critical events. Resource management, including budgeting, requires an understanding of how development resources are being utilized throughout the system, what is in the backlog of desired capabilities, and areas where there is excess capacity or capacity is insufficient for the projected demand. Budgeting is also a factor in determining how much demand is realistic regardless of capacity. Strategic changes to the resource mix across the system of systems may be needed to improve flow. Such actions are often made through hiring, contracting, or reallocation of resources between organizations and teams.

6.1.2 Executive/Stakeholder Management Visualization

6.1.2.1 Kanban Board

Backlog (Demand)	Capability Analysis		Capability Development		Done

Figure 10. ESM Kanban Template

6.1.2.2 Dashboard

Figure 11 shows a dashboard-type visualization for the ESM level.

ESM Dashboard Capabilities in Progress	CoS	Value	Total # of Requirements	# Requirements Completed Last Month	 This Month	% Value completed	# Requirements in Progress	% Value in Progress	% Requirements with work items blocked	Expected Completion
Capability 1										
Capability 2										
Capability 3										
Capability 4 (CRITICAL)										
Capability 5										

ESM Backlog	Items in backlog	
	CoS	Value
Capability 6		
Capability 7		
Capability 8		
Capability 9		
Capability 10		
Capability 11		
Capability 12		
Capability 13		

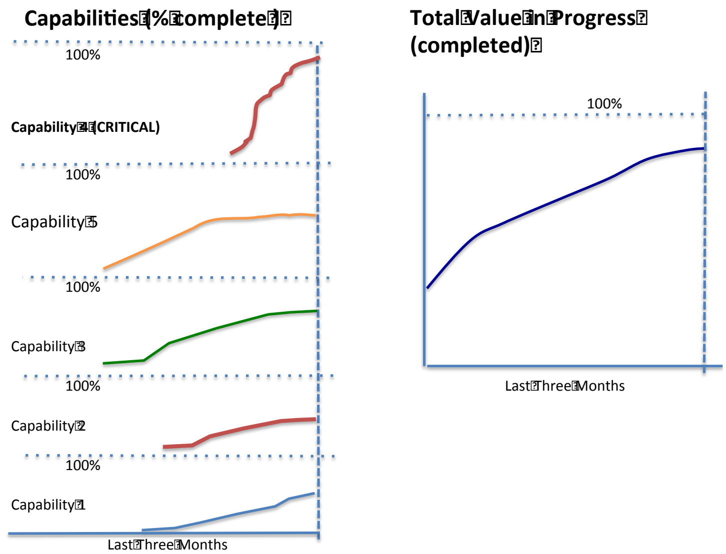


Figure 11. Notional Executive/Stakeholder Management dashboard

6.2 Capability Engineering (CE) KSS

The CE KSS represents multiple levels of activity and as the complexity grows may choose to break into multiple KSSs. However, the initial concept is a single KSS that handles a variety of different activities. First and foremost, the CE must respond to the ESM requests for both analysis and SE support to ESM decision activities and for the development of capabilities that are the highest priority to the SoS. On a secondary note, the CE provides SoS analysis support to the various PDE Teams and manages the limited number of SoS specialty engineering resources. Given the goals associated with this level, both the kanban board and the dashboard will be somewhat "busy" in terms of information.

Table 6. CE KSS Template

Capability Engineering KSS			
Demand:			
Work sources	ESM, PDT, Internal		
KSS Resources:			
Dedicated	SoS SEs, Specialist SoS SEs (performance, algorithms, interface, security…)		
Shareable	Most		
Sourced	PDE Teams		
Managed resources	Specialty SoS SEs (performance, algorithms, interface, security…)		
Activities:			
Description	*WIP Limit*	*Resource Type*	*Cohesion*
Capability Analysis		X-discipline team	Interruptible

Operational Concept Development		Internal, X-discipline team	Interruptible
Capability Requirements Creation		Internal, X-discipline team	Interruptible
Capability Requirement Development		Sourced	Interruptible
Special Engineering Services		Internal (managed)	Interruptible
Flow and Visibility:			
Additional CoS handled	Software Service CoS: One of the issues identified by the health care software developers was the amount of time that tasks were blocked waiting for SE support. For that reason, all SE Specialty Engineering work item requests from Product Teams with significant software components are assigned this Class of Service, and take on the attribute of uninterruptible. The CoS provides a guaranteed percentage of WIP capacity to be available for these requests. Resource reallocation is allowed to meet this CoS.		
Additional CoS introduced	None		
Work Selection Value Adjustments			
Source-based	*CoS-based*	*Resource-based*	*Completion-based*
			Support to work associated with requirements or capabilities within 15% of completion are raised by 10% at selection cadence points
Goals	G1. Cost-effective and timely alternatives identified for new capabilities/capability enhancements G2. Adaptable, flexible, multi-purpose solutions provided for new capabilities/capability enhancements G3. Specialty engineering responses to software teams' SE requests do not create excessive delays in capability development G4. Provide quick response to changing needs and priorities		
Questions answered	Q1. What work is currently blocked? Q2. What is the % of capabilities that are deployed within the desired timeframe? Q3. What is the predicted time to completion for "accepted" CE tasks (by class of service)? Q4. Where is capacity not meeting demand (by capability specialty engineering discipline)? Q5: Where is there excess capacity (by capability specialty engineering discipline)? Q6: What is the age of items in the CE backlog queues? Q7. What are the current CE WIP levels? Q8. What are the current CE backlog levels? Q9. What is the balance between CE WIP and CE backlog?		
Data maintained/used	KSS1: Number/status of tasks in product-level queues (initial analysis, backlog, WIP, blocked) KSS2: Number of tasks in product-level queues that are blocking other tasks (e.g., dependent tasks) KSS3: Relationships between capabilities, requirements, and features at product level KSS4: Percentage of each in-process requirement already completed/deployed KSS5: Average User Support request task completion time		
Information shared	Requirements allocation, status and deployment data; CE and PDE flow information		

6.2.1 Capability Engineering KSS Processes

6.2.1.1 Accepting/Selecting Next Work Item

As requests come in for systems engineering services, whether front end work on new capabilities or work supporting other disciplines in their developing or sustaining activities, they are accepted, roughly estimated, possibly broken into smaller tasks, and valued. An additional CoS is assigned as necessary and then the work items are added to the backlogs for the appropriate resource. Queue length limits are usually maintained for backlogs, and the level of the queue in terms of a percentage is a reasonable measure of demand. If the selection cadence is longer than daily, a WIP-limited "ready" queue can be added that allows the team to select a fixed number of tasks to accept and remain in the ready queue so that the work can be started immediately upon resource availability.

6.2.1.2 Allocating Resources and Team Development

In the case of the CE KSS, many of the tasks will require a team with expertise in one or more specialty engineering areas. In addition, for the exploration of alternative solutions, the CE team may require collaborative support from one or more PDE Team SEs. The CE must negotiate with the appropriate teams or for the specific resources they need. CoS, nearness to completion of the requirement and other factors will be considered. In the case of requests from software teams, the CoS is changed to the special software CoS guaranteeing a certain priority in handling. These tasks should be considered non-interruptible, and should be immediately returned to the source upon completion.

Capability Requirements Development tasks should be created, sourced to the various PDE Teams, and tracked to completion. Any negotiation required should be accomplished before CE or the PDE Team accepts the work.

6.2.1.3 Completion and Disbursement

As ESM analysis work items are completed, they are delivered directly to the ESM and so identified as "done" on the ESM board as well as the CE Board. Analysis tasks from PDTs are handled the same. Work sourced to the PDE Teams may be completed and then deployed by the PDE Team. The PDE Team will share data regarding its status to update the CE KSS and Dashboard. There could be an activity to provide some form of requirement completion verification and validation within the CE KSS, but in this initial concept, this is handled within PDE. This data will also be passed to the ESM KSS and dashboard.

6.2.1.4 KSS Review

Walking the CE KSS involves tracking the work in progress, identifying flow problems and blockages, resolving resource issues and blockages, and monitoring the demand queue so that when resources are available the next most valuable piece of work is accepted. The review should also track the WIP-level and demand for specialty resources to avoid blockage, overwork, or underutilization. Work items should be

scanned for adjustment to work value or priority due to achieving completion-based criteria. Technical or PDE Team issues should be reviewed. It may be appropriate to include representatives from critical PDE Teams in the review.

6.2.2 Capability Engineering Visualization

6.2.2.1 Kanban Board

The CE kanban board is divided into two parts. This top part represents the value stream for the activities that SE performs. The bottom half tracks the specialty engineers and provides the ability to monitor their work in progress for tracking work and for avoiding overtasking any single resource.

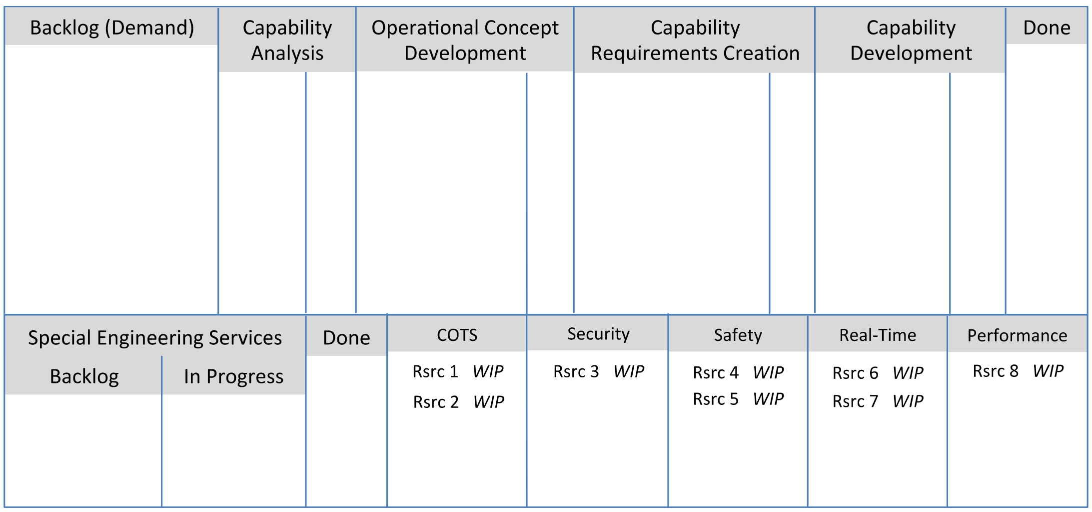
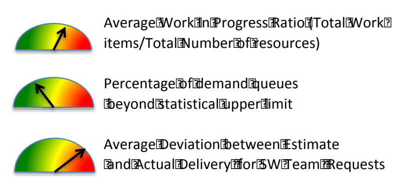


Figure 12. Notional CE Kanban Board

6.2.2.2 Dashboard

CE Dashboard			Work Items Completed					
Key Requirements in Progress	CoS	Value	Last Month	This Month	% Work Items Completed	% Value Completed	Number of work items blocked	Expected Completion
Requirement 1								
Requirement 2								
Requirement 3								
Requirement 4								
Requirement 5								
Requirement 6								
Requirement 7								
Requirement 8								
Requirement 9								
Requirement 10								
Requirement 11								
Requirement 12								
Requirement 13								
Requirement 14								
Requirement 15								



Special Eng.	Average WIP	Additional Information		
Specialty 1				
Specialty 2				
Specialty 3				
Specialty 4				
Specialty 5				
Specialty 6				
Specialty 7				

Figure 13. Notional CE Dashboard

6.3 User Support (US) KSS

User and site support personnel interact directly with the users and other operational stakeholders for the system of systems. They provide insight and triage for user

requests; they aggregate and categorize desired capabilities or required maintenance actions, and forward them for resolution to the CE or PDE Teams as appropriate. The KSS is set up to manage the resources of the personnel handling the triage function and to identify critical issues rapidly. They track issues to completion and support information requests on the status of specific issues.[*]

Table 7. User Support KSS Template

User Support KSS			
Demand:			
Work sources	User requests		
Resources:			
Dedicated	Help Desk Personnel, SW/System Engineers		
Shareable	None		
Sourced	PDE Teams, CE		
Managed resource specialties	SW/System Engineers may be handled as managed resource specialists		
Activities:			
Description	*WIP Limit*	*Resource Type*	*Cohesion*
Call Reception and triage		Internal	Must complete
Secondary ticket review		Internal	Interruptible
Ticket assignment		Internal	Interruptible
Flow and Visibility:			
Additional CoS handled	None		
Additional CoS introduced	None		
Work Selection Value Adjustments			
Source-based	*CoS-based*	*Resource-based*	*Completion-based*
None	None	None	None
Goals	**Not yet addressed**		
Questions answered	**Not yet addressed**		
Data maintained	**Not yet addressed**		
Information shared	**Not yet addressed**		

6.3.1 User Support KSS Processes

6.3.1.1 Accepting/Selecting Next Work Item

User Support functions as a clearinghouse for direction into and out of the development system from the user population. Many activities do not require development and are managed through the US KSS but do not interact with the other development KSSs. Tickets from user calls for problems that require technical development work are written up by help desk resources before they are entered into the KSS demand queue. Initial estimations are of the “t-shirt size” variety (S, M, L, XL) and the tickets are classified according to the product, domain or other attribute. Any tickets that may be critical to patient safety or other expedited activity are immediately handed off to the ESM, CE, and PDE teams to swarm and resolve quickly.. For non-critical items, initial classes of service are established along lines that support the ESM service level agreements or individual projects.

[*] The KSS for User is modeled on the system developed by Joshua Bloom at The Library Corporation, and the authors appreciate his support in this research.

6.3.1.2 Allocating Resources and Team Development

Once a ticket is entered into the demand queue, it is either determined to be product specific and forwarded to a PDE team for resolution, it is determined to involve multiple products/domains and is entered into the ESM needs backlog as a systems of systems capability issue to be considered in context with other improvements or needs, or, it is not immediately understood and so forwarded to the SoS team to analyze and recommend a handling strategy. All such tickets are maintained in the KSS as in-process and tracked through the system to completion so that User Support can provide feedback to its status to the users.

6.3.1.3 Completion and Disbursement

Upon notice from the PDE Team or CE that the development work is completed, the User Support Team advises the ticket requestor(s) that the ticket has been resolved and provides a resolution to the user. This could be in the form of a software patch or workaround. Or it could be an indication of when the fix will be deployed.

6.3.1.4 KSS Review

KSS review may be more focused on the ability to effectively triage and assign tickets. Surveillance of the status of the technical work that entered through the US KSS provides a measure of response time to user requests and may be accompanied by user satisfaction information.

6.3.2 User Support Visualization

6.3.2.1 Kanban Board

Because of the rapidity with which most help desk activities occur, the dashboard can actually provide as much information as a kanban board. While there still needs to be data gathering and issue tracking done, we assume the dashboard handles the tracking.

6.3.2.2 Dashboard

The online dashboard from The Learning Corporation shown in Figure 14 provides an example of how a Health Care User Support Dashboard might look.

Figure 14. Help Desk Dashboard from The Learning Corporation

6.4 Product Team (PT) KSS

The PDE Product Teams are responsible for one or more of the Health Care System products. The teams include systems engineers, specialty engineers, software engineers, hardware engineers, and often subject matter experts that support feature determination and development. System of system capabilities may require multiple product teams to create or enhance features, implement similar features in different ways, or collaborate to develop a common solution for the specific systems. If CE is the heart of the system of systems, the product team is the arms and legs.

The PT KSS is focused on maintaining the product at a high level of effectiveness and evolving it to support system capabilities as well as product capabilities. There is always some tension among the new feature development, older feature enhancement, and typical maintenance that is required in a technology and safety critical environment. The KSS uses the various CoS that have been defined for the system to manage flow so that response to major capability developments proceed at a reasonable pace without significant impact on ongoing project level work

Table 8. Product Team KSS Template

Product Team	
Demand:	
Work sources	US, CE, Internal, other PDE Teams

Resources:			
Dedicated	SEs, HW and SW developers		
Shareable	SEs		
Sourced	SW Developers (SDPT), SoS Specialty Engineers (CE), Domain Specialists (DPT)		
Managed resource specialties	Varies by team		
Activities:			
Description	*WIP Limit*	*Resource Type*	*Cohesion*
Requirements analysis and feature definition		Internal, X-discipline team	Interruptible
Feature development and integration		Internal, Sourced	Interruptible
Requirements V&V		Internal, Sourced	Interruptible
Deployment		Internal, Sourced	Must complete
Flow and Visibility:			
Additional CoS handled	Software Service CoS: One of the issues identified by the health care software developers was the amount of time that tasks were blocked waiting for SE support. For that reason, all SE Specialty Engineering work item requests from SW Product Development Teams are assigned this Class of Service, and take on the attribute of uninterruptible. The CoS provides a guaranteed percentage of WIP capacity to be available for these requests. Resource reallocation is allowed to meet this CoS.		
Additional CoS introduced	Certification required – Applies where work may need to be bundled to prevent costly recertification. Work selection may be accelerated or delayed when other related work items are in the demand queue.		
Work Selection Value Adjustments			
Source-based	*CoS-based*	*Resource-based*	*Completion-based*
Varies by team	Varies by team	Varies by team	Support to work associated with requirements or capabilities within 15% of completion are raised by 10% at selection cadence points
Goals	G1. Capability-allocated requirements are developed and deployed according to their product value G2: Product requirements and features are allocated to increments and spins based on value G3. Product team responds quickly to changing product needs and priorities G4. Minimize workflow disruptions in product increments and spins G5. Minimize rework due to poorly understood capability requirements G6. Product team provides timely responses to user support issues/problems		
Questions answered	Q1. Value of product-level work currently blocked? Q2. What is the % of requirements completed within the desired timeframe? Q3. Where is PT capacity not meeting demand? Q4: Where is there excess PT capacity? Q5: How often is the average age of items in product-level backlog queues outside expected levels? Q6. What are the current product-level WIP levels? Q7. What are the current product-level backlog levels? Q8. What is the product-level response time to SW requests?		

Data maintained	KSS1: Flow data on Product Team* KSS2: Number/status of tasks in demand queues KSS3: Number of tasks in product-level activities that are blocking other tasks (e.g., dependent tasks) KSS4: Relationships between capabilities, requirements, and features at product level KSS5: Percentage of each in-process requirement already completed/deployed KSS6: Average User Support request task completion time **Includes CFD (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data* .
Information shared	Flow data on Product Team*

6.4.1 Product Team KSS Processes

6.4.1.1 Accepting/Selecting Next Work Item

Selection at this level is all about balancing: the capacity with the demand, new work with ongoing activity, and SoS value with product value. While selection decisions are supported by the inherited value determination and CoSs, the product teams still negotiate the flow. The sourcing customers and PT members look at the mix of tasks in the demand queue, evaluating each according to the system values, product values and resources available, as well as considering what items represent the final parts of a requirement or capability.

All teams will implement their selection strategy to match their own need for flow control. If there are numerous smaller work items that need to be considered, a limited-size "Ready" queue could be used to provide a slower selection cadence (perhaps weekly) without impeding internal workflow while allowing decisions to be negotiated with greater analysis.

6.4.1.2 Allocating Resources and Team Development

Most of the PT work is performed by groups of resources, often in a multi-discipline project team. Individual SE resources must also be available to participate in the cross-discipline/cross-system teams used in the CE in capability analysis, so there may be a reason to apply some sort of Project-level CoS that reserves some capacity for supporting those activities. Resource allocation and tracking strategies would differ from team to team depending on the availability of scarce resources and the mix and demand for specialty resources under their control.

6.4.1.3 Completion and Disbursement

Since PTs are responsible for integration, V&V and deployment, their kanban board addresses these activities. Data on status, acceptance and availability for inclusion of the various work items in completing capability implementation is always provided upstream to the sourcing KSS.

6.4.1.4 KSS Review

Walking the kanban board and reviewing the dashboard at the product level consists of looking for blocked work—resource conflict issues, sourcing delays, and rework are the main sources here. If the PT cannot complete work items within the statistical boundaries established over time, changes must be made quickly to balance upstream expectations with performance.

6.4.2 Product Team Visualization

6.4.2.1 Kanban Board

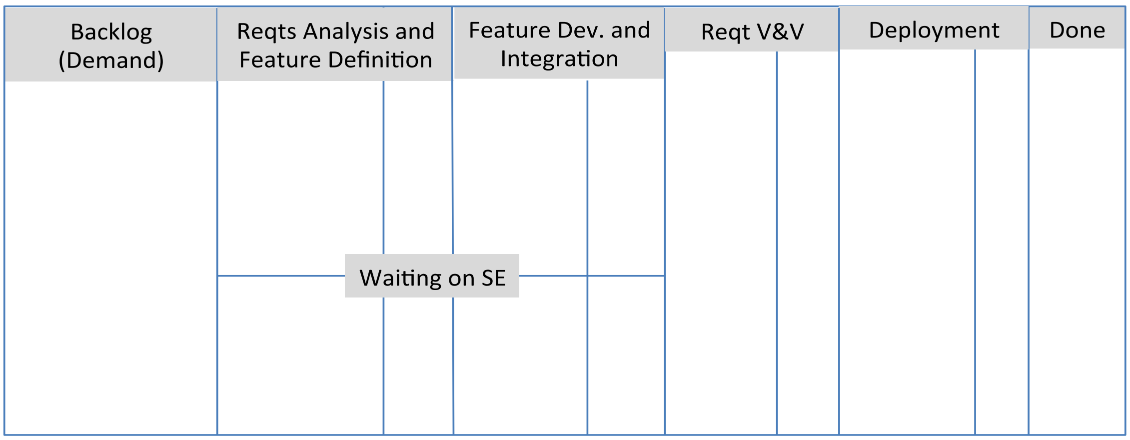


Figure 15. Notional PT Kanban Board

6.4.2.2 Dashboard

PT Dashboard			% Features	% Value	# Features	Expected
Requirements in Progress	**CoS**	**Value**	**Completed**	**Completed**	**blocked**	**Completion**
Requirement 1						
Requirement 2						
Requirement 3						
Requirement 4						
Features in Progress	Cos	Value	Reqt	Sourced to	Blocked?	
Feature 1						
Feature 2						
Feature 3						
Feature 4						
Feature 5						
Feature 6						
Feature 7						
Feature 8						
Feature 9						
Feature 10						
Feature 11						

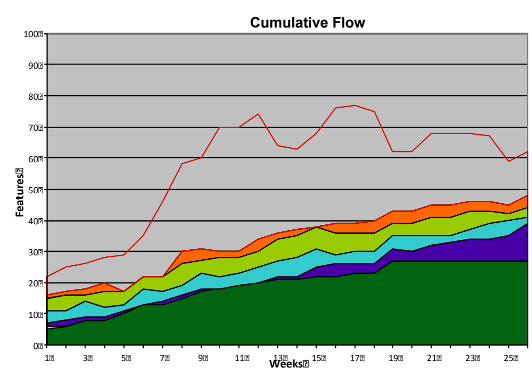


Figure 16. Notional PT Dashboard

6.5 Software Development Product Team (SPDT) KSS

The use of pull techniques in software product development has been broadly and successfully applied. However, the amount of SE activity and information provided at this level has less exposure and is significant in the Health Care scenario. Much of the

performance reporting at the capability level is dependent on the WIP, WIP limit adjustments, lead times measured, statistical limits established, and process improvement activities in the software development activities. The SPDT requests to CE for SoS SE help are one reason that the multi-level kanban and the SE as a Service concept are being considered. SE or other specialist services may require negotiations based on schedule, quality and value. A 30 day effort is worthless if it delays the product beyond its *must complete by* date.

Table 9. Software Development Product Team KSS Template

Software Development Product Team			
Demand:			
Work sources	US, PDE PTs, PDE DTs, CE, Internal		
Resources:			
Dedicated	SW SEs, Software Developers		
Shareable	All		
Sourced	SoS SE Specialty Engineering (CE), SE Specialty Engineering (PDE PTs), Domain Specialists (PDE DTs)		
Managed resource specialties	Varies by team		
Activities:			
Description	WIP Limit	Resource Type	Cohesion
Feature Analysis and Story Development		Internal, Sourced,	Interruptible or Must complete
Story Analysis		Internal, Sourced	
Story Development, Test and Integration		Internal	
Feature Integration and Test			
Flow and Visibility:			
Additional CoS handled	Certification required – Applies where work may need to be bundled to prevent costly recertification. Work selection may be accelerated or delayed when other related work items are in the demand queue.		
Additional CoS introduced	Software Service CoS: One of the issues identified by the health care software developers was the amount of time that tasks were blocked waiting for SE support. For that reason, all SE Specialty Engineering work item requests from SW Product Development Teams are assigned this Class of Service, and take on the attribute of uninterruptible. The CoS provides a guaranteed percentage of WIP capacity to be available for these requests. Resource reallocation is allowed to meet this CoS.		
Work Selection Value Adjustments			
Source-based	*CoS-based*	*Resource-based*	*Completion-based*
Varies by team	Varies by team	Varies by team	Support to work associated with requirements or capabilities within 15% of completion are raised by 10% at selection cadence points
Goals	G1. Features are developed and deployed according to their product value G2: Stories are allocated to teams and completed based on value G3. SW team responds quickly to changing product needs and priorities G4. Minimize workflow disruptions in SW development flow G5. Minimize rework due to poorly understood features and stories G6. SW team provides timely responses to issues/problems		

Questions answered	Q1. Value of software work currently blocked? Q2. What is the % of stories and features completed within the desired timeframe? Q3. Where is SW development capacity not meeting demand? Q4: Where is there excess software capacity? Q5: How often is the average age of items in software backlog queues outside expected levels? Q6. Are the current SW activity WIP levels correct? Q7. Are the current SW backlog levels as expected? Q8. What is the average external response time to SW requests?
Data maintained	KSS1: Flow data for SW Team* KSS2: Number/status of tasks in team demand queues (initial analysis, backlog, WIP, blocked) KSS3: Number of tasks blocked by other tasks (e.g., dependent tasks) KSS4: Relationships between capabilities, requirements, features and stories at SW level KSS5: Percentage of each in-process feature already completed/deployed KSS6: Average User Support request task completion time **Includes CFD (throughput, WIP, Lead time), backlog level, resource utilization, blocked tasks, and similar data.*
Information shared	Flow data, status of stories and features, specialty response measures

6.5.1 Software Development Product Team KSS Processes

6.5.1.1 Accepting/Selecting Next Work Item

SDPT work selection is more straightforward than some of the other KSSs. The highest value of work based on the combined CoS, initial value and adjusted value generally drive the selection. Where there are larger teams with more complex features, there may be specific CoSs developed to support certain types of work. For example, the amount of WIP delegated to rework (work that is caused by errors found in testing or integration) could be higher or a new CoS identified. The goal is to increase the value of the work through the team and ensure critical activities receive immediate correction.

6.5.1.2 Allocating Resources and Team Development

There are many ways to allocate resources at the SDPT level including personal selection, team selection, next available resource, or resource leveling. WIP limits control the amount of work allowed to exist within a particular activity or for a particular resource. Swarming can help to eliminate blocked work or resource-limit bottlenecks.

6.5.1.3 Completion and Disbursement

Since software is usually developed in a iterative fashion with testing and integration included as an integral part of the development activity, the completion and disbursement is reasonably straight forward. Usually the software is deployed on a test environment and validated against the story and feature requirements as well as any regression tests. Once through that hurdle the software is delivered to the Product or Domain Team for final validation and included in whatever deployment process is established for the specific requirement.

6.5.1.4 KSS Review

Most SDPTs walk the kanban board daily to identify issues, check status, and discover blocked work and bottlenecks. Resources may be reallocated during the walkthrough to address significant issues or critical expedite work.

6.5.2 Software Development Product Team Visualization

The software product team visualization in Figure 17 is a fairly standard example of software development kanban techniques from the literature (similar to Figure 2) and so holds less interest for the researchers. An important factor, however, is the CoS for SE requests that are preventing work on a particular work item to move forward.

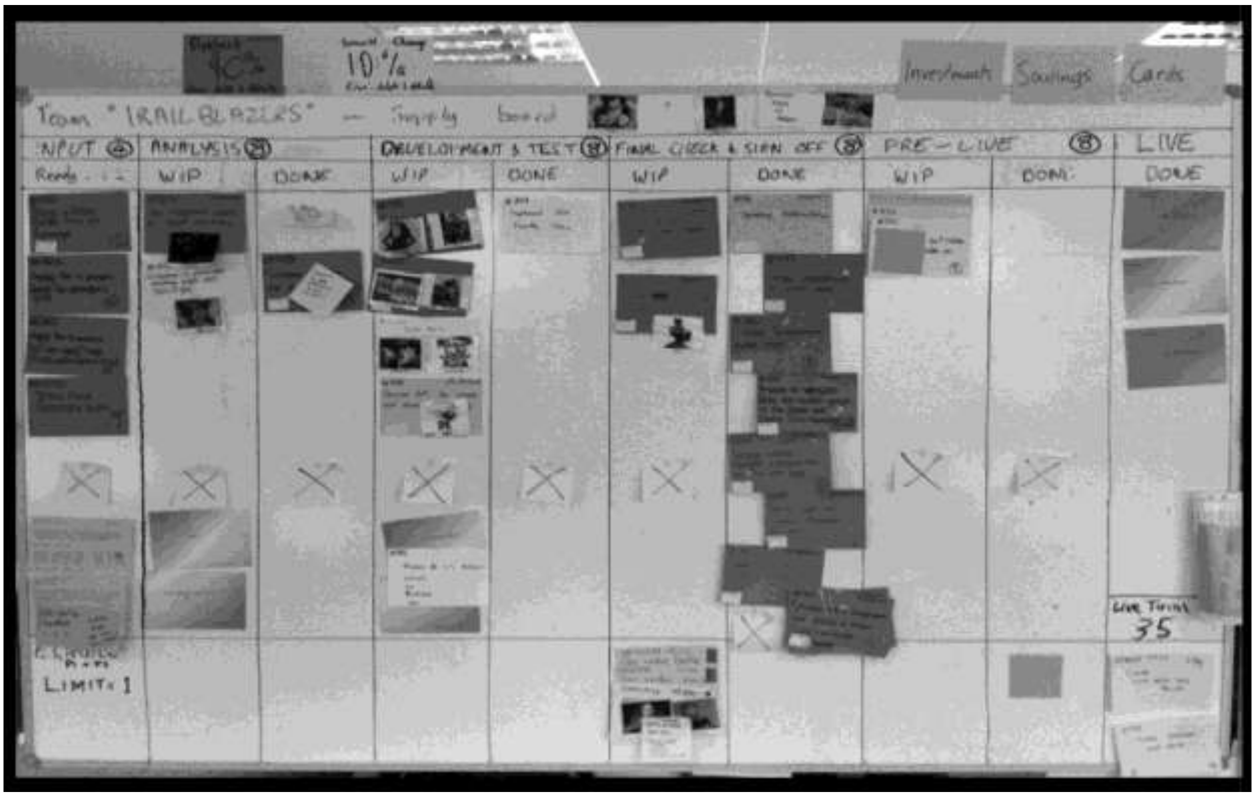


Figure 17. Possible product team visualization [Thanks to Ian Carroll]

6.6 Domain Team KSS

Domain Teams deal with cross-cutting technologies or considerations and support the CE, PDE PTs and SDPTs in handling specific areas. Examples include Database Structure or Data Definition, Network and Hardware Management, Patient Safety, Information Security (including HIPAA), and Pharmaceutical Management. These teams provide a variety of services including development, consultation, maintenance of data and information, auditing, and certification. Because each team deals with totally different activities, there is no generic way to describe them. We have provided example Dashboards and Kanban Boards as a way to show the various ways the KSSs could be implemented.

Table 10. Domain Team KSS Template

<table>
<tr><th colspan="5">Domain Team KSS</th></tr>
<tr><td>Demand:</td><td colspan="4"></td></tr>
<tr><td>Work sources</td><td colspan="4">US, Internal, PDE Teams, CE</td></tr>
<tr><td>Resources:</td><td colspan="4"></td></tr>
<tr><td>Dedicated</td><td colspan="4">Domain engineers</td></tr>
<tr><td>Shareable</td><td colspan="4">Domain engineers</td></tr>
<tr><td>Sourced</td><td colspan="4">Other PDE Teams or SDPTs</td></tr>
<tr><td>Managed resource specialties</td><td colspan="4">Domain engineers</td></tr>
<tr><td>Activities:</td><td colspan="4"></td></tr>
<tr><td colspan="2">Description</td><td>WIP Limit</td><td>Resource Type</td><td>Cohesion</td></tr>
<tr><td colspan="2">Varies by team</td><td></td><td>Varies by team</td><td>Varies by team</td></tr>
<tr><td>Flow and Visibility:</td><td colspan="4"></td></tr>
<tr><td>Additional CoS handled</td><td colspan="4">Varies by team</td></tr>
<tr><td>Additional CoS introduced</td><td colspan="4">Varies by team</td></tr>
<tr><td colspan="5">Work Selection Value Adjustments</td></tr>
<tr><td>Source-based</td><td colspan="2">CoS-based</td><td>Resource-based</td><td>Completion-based</td></tr>
<tr><td>Varies by team</td><td colspan="2">Varies by team</td><td>Varies by team</td><td>Support to work associated with requirements or capabilities within 15% of completion are raised by 10% at selection cadence points</td></tr>
<tr><td>Goals</td><td colspan="4">Varies by team</td></tr>
<tr><td>Questions answered</td><td colspan="4">Varies by team</td></tr>
<tr><td>Data maintained</td><td colspan="4">Varies by team, but includes flow measures at a minimum</td></tr>
<tr><td>Information shared</td><td colspan="4">Varies by team, but includes flow measures at a minimum.</td></tr>
</table>

6.6.1 Domain Team KSS Processes

6.6.1.1 Accepting/Selecting Next Work Item

As with any KSS, work is selected by the value. Value is usually derived from the source or the requirement-capability requesting the service. Selection is based on value, estimated effort, and available resources.

6.6.1.2 Allocating Resources and Team Development

Resources are allocated by need and by estimated effort. Swarming may occur where deemed a reasonable approach to a flow blockage.

6.6.1.3 Completion and Disbursement

For pure services, the work is over and the resources freed for further assignment. If there has been development work performed, there may be a validation activity.

6.6.1.4 KSS Review

Walking the kanban board has the same purpose as other KSSs: to identify issues and improve flow. The type of Domain team will determine the cadence for the walk, although a daily review is probably a reasonable goal.

6.6.2 Domain Team Visualizations

The Domain Teams visualizations must be suited to the particular domain. The examples of kanban boards and dashboards shown in the previous sections provide some examples of the type of information displayed.

6.7 Flow of Information between the KSSs

Figure 18 shows how the data flows between KSS kanban boards and dashboards.

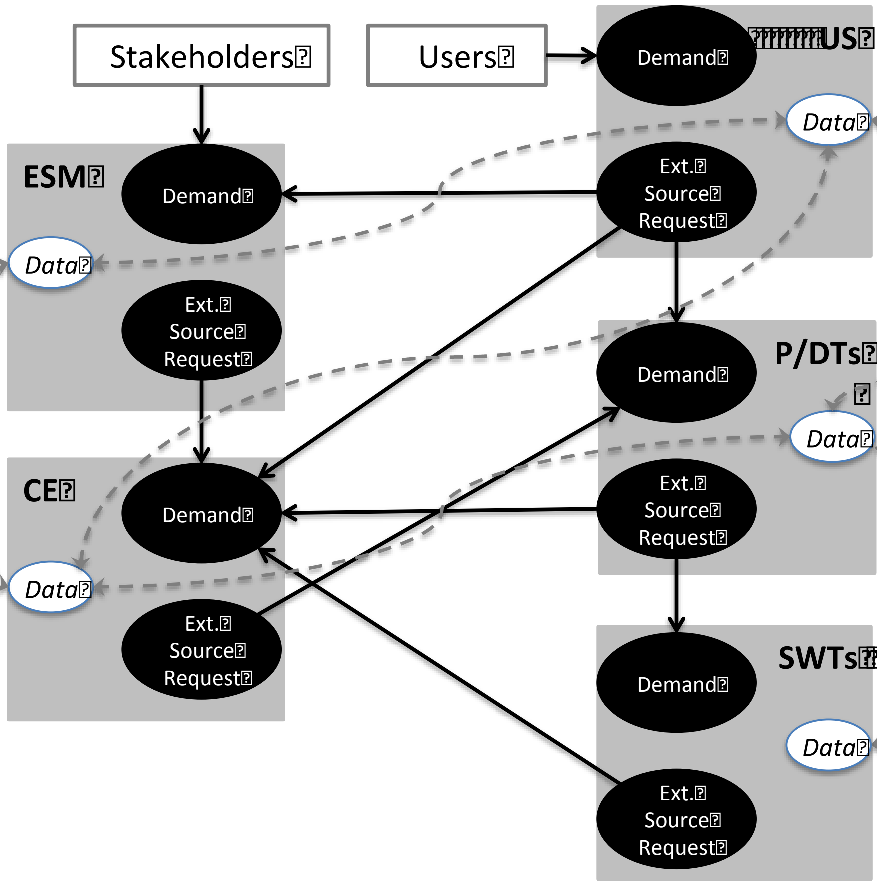


Figure 18. Data and external source requests flow between KSSs

7 Using The Health Care KSS Network

7.1 Introduction

In this section, we provide an example of how the KSS Network might be employed for the evolution and maintenance of the health care SoS described in Section 4 of this report.

7.2 Current New Capability, Upgrade, and Enhancement Plans

The following describe several new capabilities that need to be implemented within the health care SoS. In this example, there is already an existing backlog into which these new requests are inserted. The insertion of these new requests cause a re-assessment and potentially a re-prioritization of the tasks in the existing backlog. For example, lower priority features may be deferred to a later increment/release.

7.2.1 New capability to interface to a new health insurance company

The health insurance company has a claims form that is much more detailed than any of the other health insurance forms currently supported by the system. The new claims systems will require the health care organization to capture additional information about patients, diagnoses, and physician orders (tests, patient monitoring, and prescriptions) than is currently maintained in the system.

7.2.2 New capability to integrate and analyze information from multiple patient telemetry systems to improve diagnostic capabilities

At least two alternative approaches need to be evaluated before software development can begin:

1. Determine if there are any COTS products currently in the marketplace that can perform the needed data fusion. If candidate COTS products can be identified, the task is to identify and evaluate these COTS products and select the "best" one, then integrate it into the enterprise. While relatively easy, this approach will still require inputs from the users (physicians) to define the desired telemetry inputs for integration and the desired views to be output by the system.
2. If there are no COTS products that can integrate the information from the existing telemetry systems, the tradeoff is to evaluate is whether to:
    a. change non-compatible telemetry systems for more compatible ones and use a COTS product to integrate/analyze the desired information or
    b. develop a custom application to do the integration and analysis. This approach will require more detailed inputs from the user community than

the first alternative.

7.2.3 User response improvement

Due to the growth in patients and staff, the system response time has become unacceptably slow and is potentially putting patient safety at risk. The goal of this task is to evaluate alternatives for improving the user response time and recommend one or more to the Board of Directors for funding. Options under consideration are hardware upgrades, network upgrades, and software/database refactoring.

7.2.4 Periodic upgrade of pharmacy formulary information

The pharmacy data on formularies and drug interactions is maintained through a subscription service. The service typically sends out updates quarterly. These updates must be analyzed against the existing pharmacy database structures in the health care system, any necessary updates to the data structures made, the data structure updates tested and deployed, then the data updates populated in the database. Pharmacy formulary upgrades requiring database structure modifications are deployed with the database structure upgrades.

7.2.5 Patient Safety Issue Due to Interoperability Problem (Patient Safety Crisis):

The latest update of the health care products included a feature to send patient records to an external health care system so that records could be electronically transmitted for patients transferred from one facility to another. The new feature was implemented and fully tested and seemed to function well during the first 30 days after deployment. However, late one night, a physician noticed that an important entry made by the other health care system was not showing up properly in the time log.

7.3 Capability Entry Points and Initial Triage

To understand the process flow, one needs to understand the entry points for new capabilities as well as upgrades to existing capabilities and periodic upgrades to keep information in both the SoS and the products current.

The first three capability requests (*Insurance Interface*, *Telemetry Integration*, and *Performance Upgrade*) are submitted at the SoS level through the Executive/Stakeholder Management (ESM). Requests at this level can come from SoS stakeholders through various channels, users through the User Support (US) Help Desk, or Capability Engineering (CE) engineering specialty teams identifying potential infrastructure issues that must be scheduled for remediation/upgrade. The requests are added to the Needs Backlog to await the next ESM work selection activity.

The fourth capability (*Formulary Upgrade*) is a periodic upgrade that, when received must be processed and deployed by a specified date since it affects the medications that a patient can receive as well as contraindications related to usage (a patient safety issue). The upgrade enters the system through the Pharmacy Domain Team for analysis and implementation.

The critical fifth problem was immediately reported to the Health Care Help Desk as a patient safety issue and a Critical-Expedite ticket generated to fix the problem immediately.

7.4 KSS Network Operations

The Capability Engineering (CE) team consists of several general systems engineers as well as several systems engineers with special expertise. The engineering specialties include hardware/platform sizing, performance, and configuration; Commercial-Off-the-Shelf (COTS) product assessment; networking; architecture; algorithm/data fusion/interfaces/interoperability; security; safety; and reliability. In addition, when product architecture or internal information is required to perform the *Translating Capabilities to Requirements* and *Addressing Options* SoS activities, the CE engineering team requests support from the appropriate product teams.

7.4.1 Executive/Stakeholder Management and Capability Engineering Operations

When the three new SoS capability requests (*Insurance Interface*, *Telemetry Integration*, and *Performance Upgrade*) are evaluated by the ESM, they are either approved for development or approved for an initial assessment. In either case, associated tasks are placed in the CE KSS demand backlog and assigned a CoS, an estimated size and date of completion, and an indication of the engineering specialties that are required to work the task:

> *Insurance Interface* is approved for implementation and assigned a **Standard** CoS. Further analysis by the CE team indicates that it requires Interface/Interoperability resources.
>
> *Telemetry Integration* is assigned a **Standard** CoS and an initial CE assessment that includes an analysis of alternatives is initiated. The analysis of alternatives includes a COTS product assessment and the development of a specification for algorithm, data fusion, interfaces, and interoperability requirements.
>
> *Performance Upgrade* is assigned an **Important** CoS and approved for an initial CE assessment and analysis of alternatives. In addition, it is identified as requiring hardware, networking, and architecture resources.

Because *Performance Upgrade* is an **Important** CoS, it will be the work item selected by the first available hardware, networking, and architecture resources. The other work becomes part of the CE KSS demand backlog.

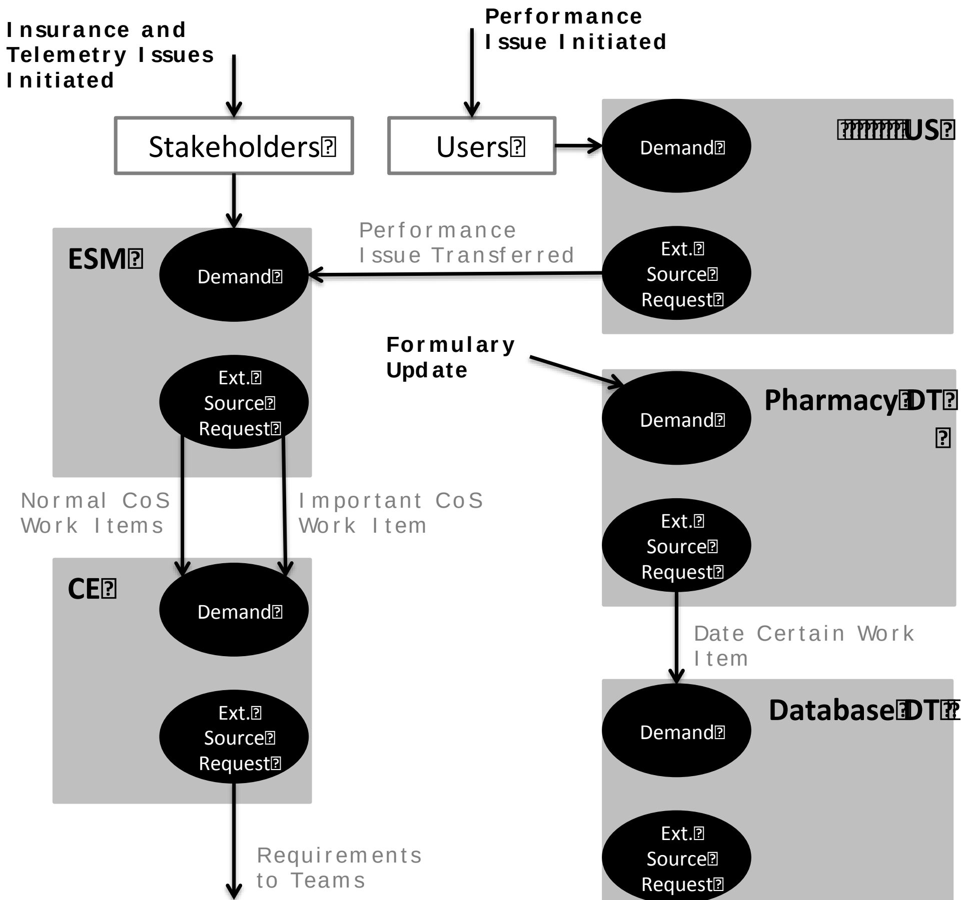


Figure 19. ESM Operations

7.4.2 Incoming Critical Expedite Capability

As soon as the Critical Expedite Patient Safety work item enters the system all teams are notified and a small high-caliber SE team from CE suspends their current activities to immediately begin work on the problem. Initial analysis shows that the candidate areas for the problem are any product that updates the electronic patient record physician entries or the new interface software.

The assessment team is expanded into a cross-discipline/cross-product team to include representatives from each of these product teams (possibly suspending more WIP). The resulting analysis finds that the problem is due to an interoperability issue with the system clocks on two systems: one indicated midnight as 24:00:00 and the external system indicated midnight as 00:00:00 the next day. The issue was not discovered until an entry was made in a patient record at midnight and that record was then transmitted to the external system. After transmission, the receiving system inserted the transmitted log entries into the new system using the date/time stamp on the entries, resulting in the midnight annotation being posted on the wrong date and almost missed by the attending physician.

Once the source of the problem is identified, the crisis response team consults with the interfacing system to explain the problem and develops a mutually acceptable solution. Investigations showed that the more standard way to represent midnight is 00:00:00, so requirements are created for the external interface handler team to convert any outgoing midnight timestamps to 00:00:00 and any incoming midnight timestamps to 24:00:00.

The team develops **Critical Expedite** work items that are created and assigned to implement, test, and deploy immediate patches to all affected sites. In addition, requirements for a longer-term permanent solution are defined and integrated into the current development release and assigned an **Important** CoS.

As team members complete their work on the Critical Expedite work items, they resume their previously suspended work items. The results to the normal KSS flow are evident in the ongoing flow measurements, and any additional changes to priority or CoS to remedy issues from the delayed work are accomplished. Work then continues as before the crisis.

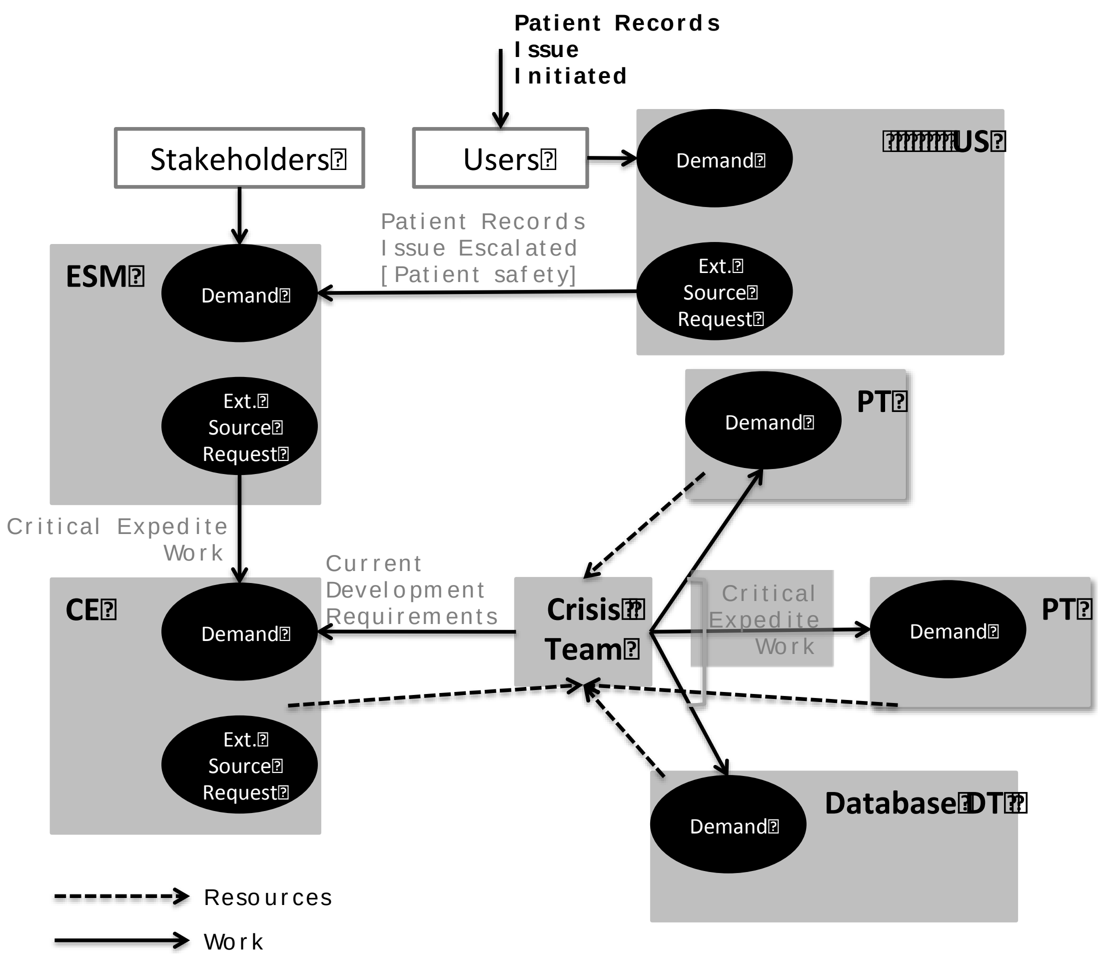


Figure 20. Critical Expedite Operations

7.4.3 Pharmacy Domain Team Kanban Operations

The pharmacy domain team expects to receive periodic updates of the formulary and drug interactions data. These updates always carry a date as to when the information changes must be implemented. Often the update only requires that the old data be removed and the new data imported to the database. However, in this case the update requires data structure changes to the database. The pharmacy team generates a work item for the Database Domain Team requesting the data structure changes. Since there are 15 days before the change must be implemented, the work item is assigned a **Date Certain** CoS. In other circumstances, the work item may be allocated an Important or even Critical Expedite CoS depending on the content of the pharmacy update.

The database team conducts an analysis of the requested changes to determine what other products might be impacted by the data structure changes. The database team then generates work items of appropriate CoS for the affected product teams to handle the database changes and determine if any additional changes are needed in their products to support the formulary database update. Once all of the related changes have been made and tested, they are installed in the next increment integration image for further integration testing and deployment.

8 Modeling and Simulation of the Prototype KSS Network

As described in the Phase I report of this research project, the overall goal of the modeling component of this task is to verify whether organizing projects as a set of cooperating kanbans (a kanban-based scheduling system, KSS) results in better project performance. Performance is measured through a value function, and better performance is defined as achieving value along one or more of the following scales, which seem most relevant to the rapid-response environment:

- Shortest-time to initial-value
- Highest-value in the quickest-time
- Highest-value for a given-time

The research question we seek to answer is: can value be improved through a KSS that controls the interaction of a resource-limited systems engineering team with one or more development teams via a service-oriented interface implemented. We hypothesize that if systems engineering produces a partial system definition (context, requirements, etc.) earlier, and releases that definition to development, the defects inherent in that less-complete definition can be resolved through coordinated, task-directed KSS interactions between development and systems engineering. In this way, the total value realized by the project within its critical availability time limits is improved over the traditional up-front and separated parallel design process.

8.1 Model Alternatives Explored

As was reported in the Phase 1 report, several alternative modeling techniques were explored for this task. The research team learned from each of these modeling attempts and attempted to refine the modeling approach as much as possible within the schedule constraints of this research effort.

Three approaches to modeling were considered for this research:

- System dynamics modeling
- Discrete-event modeling
- Agent-based modeling

Each of these modeling approaches has advantages for the problem domain and level of abstraction, as illustrated in Figure 21.

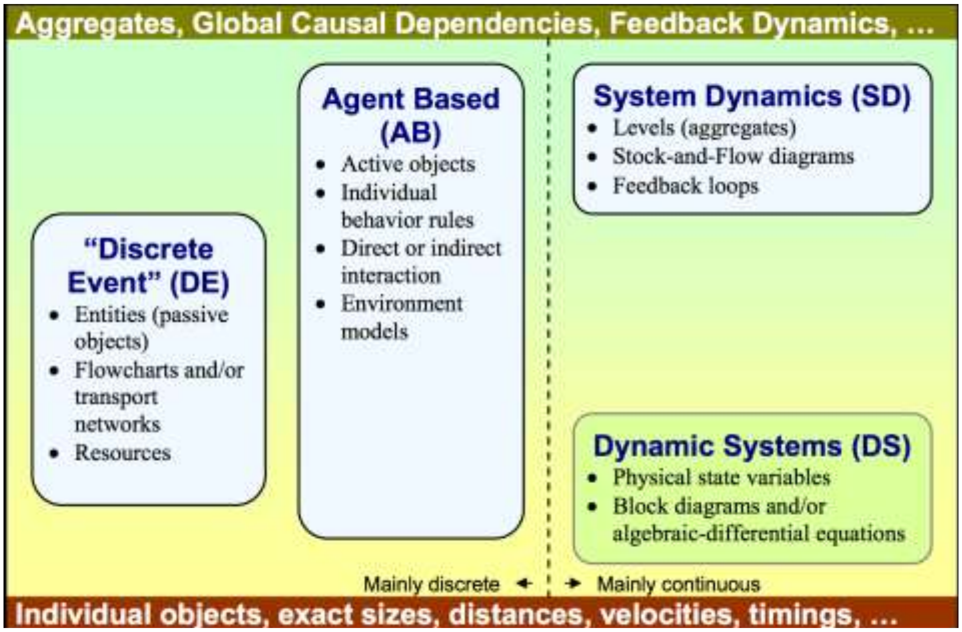


Figure 21. Modeling approach vs. abstraction level (Borshchev and Filippov 2004)

The results of this analysis of simulation alternatives indicated that a combined discrete-event and continuous model would be the best approach for modeling the KSS network and assessing its impacts on work flow at the various levels, efficiency of the engineering and development teams, and system capability "value" achieved over time. The rest of this section describes the combined discrete-event and continuous model development and lessons learned from this activity.

8.2 Combined Discrete-Event and Continuous Model

A discrete-event and continuous model was developed in the hopes of better understanding the KSS network described in this report. Discrete event entities were selected to capture individual task characteristics that are critical in an actual Kanban management scheduling process. The different priorities of the tasks are used for scheduling, and the WIP itself is managed as a discrete quantity. Individual performers are also mapped to tasks and this aspect can be modeled with discrete attributes.

There are also important continuous parameters that drive people's behavior, including perception delays, feedback effects, schedule pressure and deadlines, motivation and other management pressures. Therefore, it was felt that a combined discrete-event – continuous approach would provide a richer and more holistic perspective with interacting model compartments. The following highlights key features of the KSS network simulation:

- Continuous flows for tasks and value accumulation are driven by the discrete events. The corresponding rates are pulsed at the event times for task completion and value attainment. The aggregate accumulations are used for continuous quantities such as schedule pressure due to do progress gaps.

- The simulation model and tools codify the multi-level enterprise framework described. We have leveraged the Internet to improve accessibility and usability across a wide range of usage scenarios.
- The software architecture for a combined discrete-event and continuous simulation toolset was driven to achieve usability and interoperability with other tools, utilities and language frameworks that can be used to further instrument the simulation and analyze results.

The following describes the actual implementation of the KSS discrete-event – continuous simulation.

8.3 MODEL IMPLEMENTATION

The simulation framework models the relations between capabilities, requirements, product increments and spins in SoS enterprises. Capabilities and requirements are special work items for an enterprise. They are decomposed into smaller work items to be implemented in product increments and spins by systems and software engineering.

Capabilities are the set of desired functions for the enterprise:

$$C: \{C_1, C_2, C_3, \ldots C_J\}.$$

Each capability has attributes including the requesting source (e.g. user, stakeholder, executive management), a description, Class of Service (CoS), value, date-time authorized, and the requirements set of all requirements defined to fully implement the capability. The capability effort is a rolled up sum of labor hours for the requirements.

The capabilities are enabled through a set of decomposed requirements:

$$R: \{R_1, R_2, R_3, \ldots R_K\}.$$

Each requirement may fulfull one or more capabilities. The requirements may be SoS requirements or individual system requirements.

Requirements have attributes for the set of capabilities supported, a CoS derived from highest capability CoS, and value as a sum of capabilities supported values.

The set of products P are the separate increments of product development:

$$P: \{P_1, P_2, P_3, \ldots P_L\}.$$

These products are primarily managed independently, but not always depending on enterprise priorities and teams involved. The products are ongoing separate systems yet also contribute to overall SoS enterprise capabilities.

Each increment is then planned out as incremental spins to achieve the full increment. The model framework allows for a variable percent of requiremeents effort allocation across products.

The capabilities and requirements sets are connected in a bipartite graph relationship. All requirements can be potentially mapped to any given capability. The requirements

and products relationship is also bipartite. This tree of connections between capabilities, requirements and products is better visualized in the next section with a worked out example.

8.4 SIMULATION EXAMPLE

This example walks through the threads of a distributed simulation using the simulation tool and associated files at http://softwareprocessdynamics.org/models/se_kanban/sos_enterprise.php.

Note the simulation tool used for this is example is undergoing enhancements. The screenshots examples include a couple graphic placeholders in in the current working version.

In this example distributed refers to the spawning of multiple spins developing decomposed fine grain tasks such as use cases. Collectively they implement the requirements decomposed from the top-level enterprise capabilities.

This baseline example corresponds to the enterprise relationships shown in Figure 4 in Section 4.2, with additional value and effort associations for quantitative simulation.

We show representative inputs and outputs from elements of a simulation run. We illustrate pieces of a full simulation down to the view of an individual Kanban board showing task progress. Note this is only a single run, and multiple runs are available using Monte Carlo simulation.

The SoS relationships for this example shown in Figure 22 illustrate the allocation of three enterprise capabilities across seven requirements and implemented in four product increments. Inputs to the model begin at the highest enterprise level consisting of the numbers of new and modified capabilities.

Each requirement is assigned a total effort through its mapping to capabilities. There is much overlapping and cross interactions in the actual sponsor environment as represented in this example. Each product then rolls up these allocations to an increment.

We also model the constraints for specialty engineering resources in the workflows. The requirement color coding in Figure 22 indicates the attribute for resource type needed (e.g. hardware, network, database).

These tags are carried through as discrete event attributes for logic operations, such as checking for resources and flow decisions. The later Kanban boards shows special resource tasks that flowed down.

In this example for simplicity we assume an even 100 person-months for each of the three top-level enterprise capabilities. They are modeled as all new though our simulation framework handles modified capabilities as percentages relative to new.

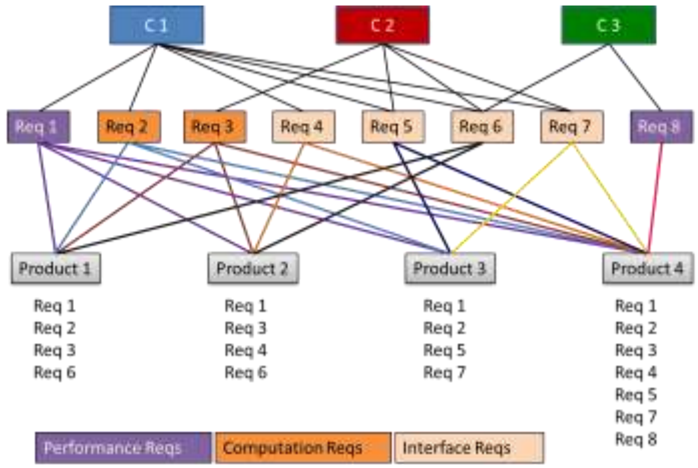


Figure 22. Example SoS Enterprise Capabilities to Requirements to Products

Also assumed is an equal allocation of the capabilities among their constituent requirements. Thus for Capability C_1 the six requirements $\{R_1, R_2, R_4, R_5, R_6, R_7\}$ equally contribute to it in person-months of effort. All capabilities in this example use an even distribution among requirements.


```xml
<?xml version="1.0" encoding="UTF-8" ?>
<simulation>

  <capability>
    <name>MC1</name>
    <requirement>
      <name>R1</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R2</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R4</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R5</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R6</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R7</name>
      <value>60</value>
    </requirement>
  </capability>
```

...

```xml
<requirement>
    <name>R1</name>
    <effort>50</effort>
    <type>Performance</type>
    <resource_type>Security</resource_type>
</requirement>

<requirement>
    <name>R2</name>
    <effort>50</effort>
    <type>Computation</type>
    <resource_type>Hardware</resource_type>
    <resource_type>Database</resource_type>
</requirement>
```

...

```xml
<product>
  <name>P1</name>
    <requirement>
      <name>R1</name>
      <effort_percent>25.0</effort_percent>
    </requirement>
    <requirement>
      <name>R2</name>
      <effort_percent>33.3</effort_percent>
    </requirement>
    <requirement>
      <name>R3</name>
      <effort_percent>33.3</effort_percent>
    </requirement>
    <requirement>
      <name>R6</name>
      <effort_percent>50.0</effort_percent>
    </requirement>
</product>
```


Figure 23 illustrates a portion of the XML input for the distributed simulation.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<simulation>

  <capability>
    <name>MC1</name>
    <requirement>
      <name>R1</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R2</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R4</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R5</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R6</name>
      <value>60</value>
    </requirement>
    <requirement>
      <name>R7</name>
      <value>60</value>
    </requirement>
  </capability>
```

...

```xml
<requirement>
    <name>R1</name>
    <effort>50</effort>
    <type>Performance</type>
    <resource_type>Security</resource_type>
</requirement>

<requirement>
    <name>R2</name>
    <effort>50</effort>
    <type>Computation</type>
    <resource_type>Hardware</resource_type>
    <resource_type>Database</resource_type>
</requirement>
```

...

```xml
<product>
  <name>P1</name>
    <requirement>
      <name>R1</name>
      <effort_percent>25.0</effort_percent>
    </requirement>
    <requirement>
      <name>R2</name>
      <effort_percent>33.3</effort_percent>
    </requirement>
    <requirement>
      <name>R3</name>
      <effort_percent>33.3</effort_percent>
    </requirement>
    <requirement>
      <name>R6</name>
      <effort_percent>50.0</effort_percent>
    </requirement>
</product>
```

Figure 23. XML Simulation Input Portions

The simulation engine parses the XML and sets up the decomposed stochastic simulations down to the spin level. For example, it rolls up effort for the Product 1 four constituent requirements mappings. The term P_iR_j denotes that product Pi fulfills requirement R_j in some degree, so the four requirements mappings for P_1 are:

$$P_1\,(R_1, R_2, R_3, R_6).$$

The Product Increment effort quantities are rolled up this way, as shown in the simulation output in Figure 24.

This baseline example shows the decomposition from requirements to product tasks. The output shows a narrative of the parsed enterprise description with the previous notations for capabilities and products; with the allocated requirements, values and effort associations.

Note the top-level "sample value output" charts are temporary partial mockups. They are being populated with aggregated rollups from the products. The product level requirements trends shown for Product P1 are actuals.

SoS Enterprise Process Simulation

Select a file: sos_enterprise_value_example.xml

model = sos_enterprise_value_example.xml

This simulation is for 3 capabilities and 8 requirements across 4 products. The total enterprise value is 619.

MC1(R1,R2,R4,R5,R6,R7) **MC2**(R3,R5,R6) **MC3**(R6,R8)

P1(R1,R2,R3,R6) **P2**(R1,R3,R4,R6) **P3**(R1,R2,R5,R7) **P4**(R1,R2,R3,R4,R5,R7,R8)

Requirement **R1** is 50 PM effort of type Performance with 1 resource types
Requirement **R2** is 50 PM effort of type Computation with 2 resource types
Requirement **R3** is 50 PM effort of type Computation with 2 resource types
Requirement **R4** is 50 PM effort of type Interface with 2 resource types
Requirement **R5** is 50 PM effort of type Interface with 2 resource types
Requirement **R6** is 50 PM effort of type Interface with 2 resource types
Requirement **R7** is 50 PM effort of type Interface with 2 resource types
Requirement **R8** is 50 PM effort of type Performance with 2 resource types

Capability **MC1** has 6 requirements
  Requirement R1 has value 60
  Requirement R2 has value 60
  Requirement R4 has value 60
  Requirement R5 has value 60
  Requirement R6 has value 60
  Requirement R7 has value 60

Capability **MC2** has 3 requirements
  Requirement R3 has value 33.3
  Requirement R5 has value 33.3
  Requirement R6 has value 33.3

Capability **MC3** has 2 requirements
  Requirement R6 has value 80
  Requirement R8 has value 80

Product **P1** develops 4 Requirements
  Requirement R1 25.0% Effort = 12.5 PM
    5 requirement tasks: R1.1, R1.2, R1.3, R1.4, R1.5,
  Requirement R2 33.3% Effort = 16.5 PM
    6 requirement tasks: R2.1, R2.2, R2.3, R2.4, R2.5, R2.6,
  Requirement R3 33.3% Effort = 16.5 PM
    6 requirement tasks: R3.1, R3.2, R3.3, R3.4, R3.5, R3.6,
  Requirement R6 50.0% Effort = 25 PM
    10 requirement tasks: R6.1, R6.2, R6.3, R6.4, R6.5, R6.6, R6.7, R6.8, R6.9, R6.10,
Total **P1** Effort = 70.5 PM, 27 Tasks
P1 Simulation with input file P1.xml

Product **P2** develops 4 Requirements
  Requirement R1 25.0% Effort = 12.5 PM
    5 requirement tasks: R1.1, R1.2, R1.3, R1.4, R1.5,
  Requirement R3 33.3% Effort = 16.5 PM
    6 requirement tasks: R3.1, R3.2, R3.3, R3.4, R3.5, R3.6,
  Requirement R4 50.0% Effort = 25 PM
    10 requirement tasks: R4.1, R4.2, R4.3, R4.4, R4.5, R4.6, R4.7, R4.8, R4.9, R4.10,
  Requirement R6 50.0% Effort = 25 PM
    10 requirement tasks: R6.1, R6.2, R6.3, R6.4, R6.5, R6.6, R6.7, R6.8, R6.9, R6.10,
Total **P2** Effort = 79 PM, 30 Tasks
P2 Simulation with input file P2.xml

…

Product **P3** develops 4 Requirements
  Requirement R1 25.0% Effort = 12.5 PM
    5 requirement tasks: R1.1, R1.2, R1.3, R1.4, R1.5,
  Requirement R2 33.3% Effort = 16.5 PM
    6 requirement tasks: R2.1, R2.2, R2.3, R2.4, R2.5, R2.6,
  Requirement R5 50.0% Effort = 25 PM
    10 requirement tasks: R5.1, R5.2, R5.3, R5.4, R5.5, R5.6, R5.7, R5.8, R5.9, R5.10,
  Requirement R7 50.0% Effort = 25 PM
    10 requirement tasks: R7.1, R7.2, R7.3, R7.4, R7.5, R7.6, R7.7, R7.8, R7.9, R7.10,
Total **P3** Effort = 79 PM, 30 Tasks
P3 Simulation with input file P3.xml

Product **P4** develops 7 Requirements
  Requirement R1 25.0% Effort = 12.5 PM
    5 requirement tasks: R1.1, R1.2, R1.3, R1.4, R1.5,
  Requirement R2 33.3% Effort = 16.5 PM
    6 requirement tasks: R2.1, R2.2, R2.3, R2.4, R2.5, R2.6,
  Requirement R3 33.3% Effort = 16.5 PM
    6 requirement tasks: R3.1, R3.2, R3.3, R3.4, R3.5, R3.6,
  Requirement R4 50.0% Effort = 25 PM
    10 requirement tasks: R4.1, R4.2, R4.3, R4.4, R4.5, R4.6, R4.7, R4.8, R4.9, R4.10,
  Requirement R5 50.0% Effort = 25 PM
    10 requirement tasks: R5.1, R5.2, R5.3, R5.4, R5.5, R5.6, R5.7, R5.8, R5.9, R5.10,
  Requirement R7 50.0% Effort = 25 PM
    10 requirement tasks: R7.1, R7.2, R7.3, R7.4, R7.5, R7.6, R7.7, R7.8, R7.9, R7.10,
  Requirement R8 100.0% Effort = 50 PM
    19 requirement tasks: R8.1, R8.2, R8.3, R8.4, R8.5, R8.6, R8.7, R8.8, R8.9, R8.10, R8.11, R8.12, R8.13, R8.14, R8.15, R8.16, R8.17, R8.18, R8.19,
Total **P4** Effort = 170.5 PM, 65 Tasks
P4 Simulation with input file P4.xml

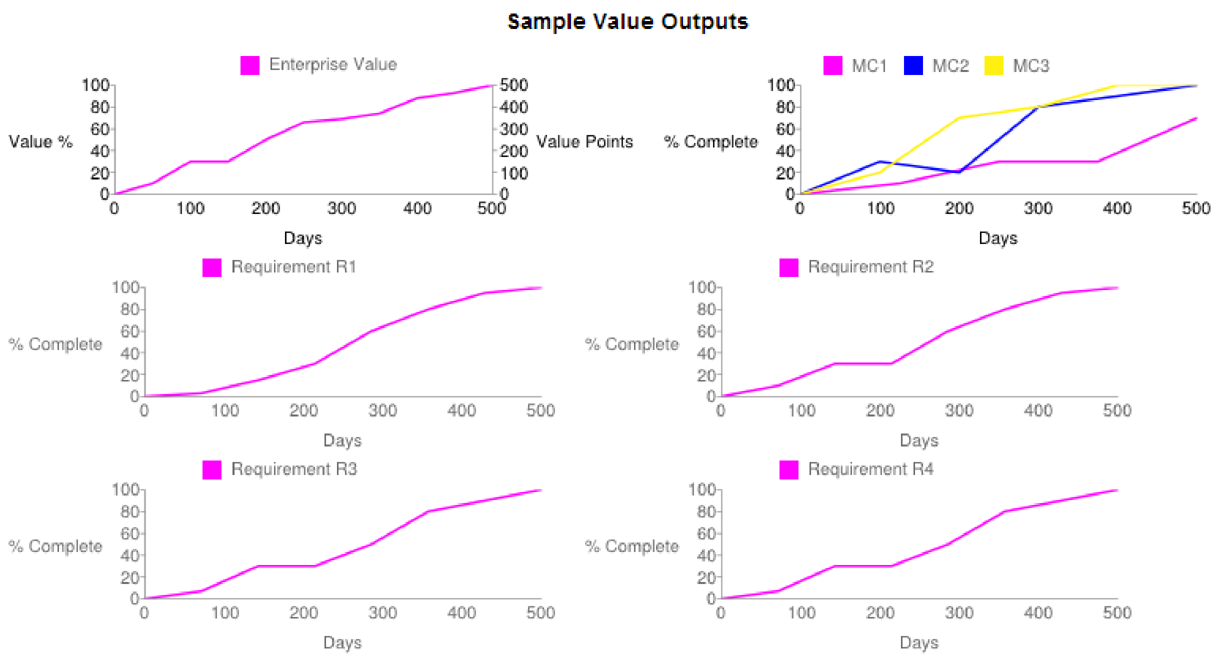


Figure 24. Example Enterprise Top-Level Simulation Output

The spin level simulations use probability distributions for the task effort expenditures. The top-level view of a spin is seen in Figure 5 summarizing the project timelines, aggregate progress and individual task events.

The simulation tool opens up the product increments in separate browser tabs with the generated product input files. Each time the simulation is run the product level files are recreated.

A portion of the generated intermediate file P1.xml is shown in Figure 25. It describes the 27 tasks fed into the product level simulation for P1. The link to this file is shown in Figure 24 for the Product P1 rollup.


```xml
<?xml version="1.0" encoding="UTF-8" ?>
<simulation>
<task>
<task_name>R1.1</task_name>
<parent_requirement>R1</parent_requirement>
<task_value>6</task_value>
</task>
<task>
<task_name>R1.2</task_name>
<parent_requirement>R1</parent_requirement>
<task_value>6</task_value>
</task>
<task>
<task_name>R1.3</task_name>
<parent_requirement>R1</parent_requirement>
<task_value>6</task_value>
</task>
<task>
<task_name>R1.4</task_name>
<parent_requirement>R1</parent_requirement>
<task_value>6</task_value>
</task>
```


Figure 25. Product P1 Generated Input File

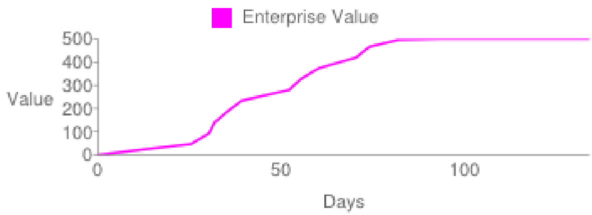
Product P1 Increment Process Simulation

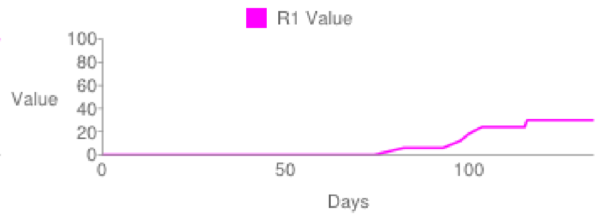
# SE Tasks (1-10)	6	# SE People (1-5)	1	SE WIP Limit (1-10)	2
# Software Tasks (5-40)	27	# Software People (1-10)	5	Software WIP Limit (1-10)	5

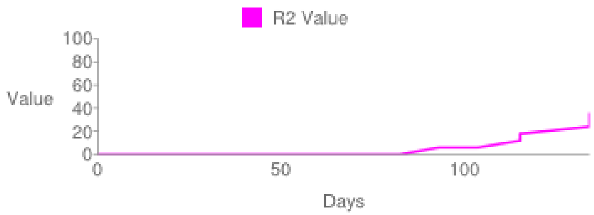
Monte Carlo Simulation Off

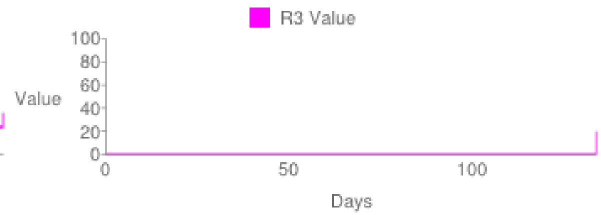
Run

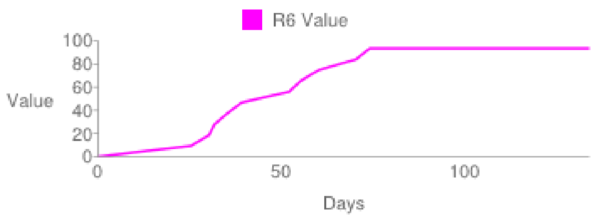
Input file = P1.xml







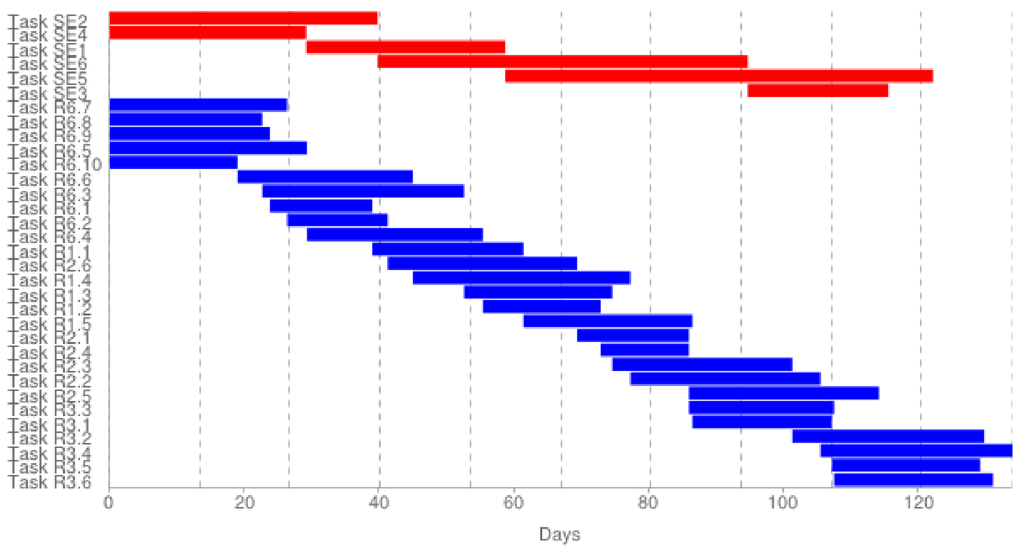




P1 Results

SE Effort = 950.3 Person-Hours (6.3 PM)
SWE Effort = 5100.8 Person-Hours (33.6 PM)
Schedule = 133.9 Days
Enterprise Value = 206.28
Productivity = 0.00529 Tasks / Person-Hour
Project Value = 179.28
Productivity Multiplier = 1

Kanban Gantt (SE WIP=2, SWE WIP=5)



5a

SWE Task	Org. Value	Project Value	Effort (Person-Hours)	Resource(s)	Duration	Start	Finish	R1 Cum. Value	R6 Cum. Value	Total Cum. Value
R6.10	9.33	9.33	151.3	1 SWE	18.9	0	18.9	0	9.33	9.33
R6.8	9.33	9.33	181.1	1 SWE	22.6	0	22.6	0	18.66	18.66
R6.9	9.33	9.33	189.9	1 SWE	23.7	0	23.7	0	27.99	27.99
R6.7	9.33	9.33	210.4	1 SWE	26.3	0	26.3	0	37.32	37.32
R6.5	9.33	9.33	233.7	1 SWE	29.2	0	29.2	0	46.65	46.65
R6.1	9.33	9.33	121.2	1 SWE	15.2	23.7	38.9	0	55.98	55.98
R6.2	9.33	9.33	118.8	1 SWE	14.9	26.3	41.2	0	65.31	65.31
R6.6	9.33	9.33	208.2	1 SWE	26	18.9	44.9	0	74.64	74.64
R6.3	9.33	9.33	238.8	1 SWE	29.9	22.6	52.5	0	83.97	83.97
R6.4	9.33	9.33	208.9	1 SWE	26.1	29.2	55.3	0	93.3	93.3
R1.1	6	6	179	1 SWE	22.4	38.9	61.3	6	93.3	99.3
R2.6	6	6	224.6	1 SWE	28.1	41.2	69.3	6	93.3	105.3
R1.2	6	6	139.6	1 SWE	17.5	55.3	72.8	12	93.3	111.3
R1.3	6	6	175.6	1 SWE	22	52.5	74.5	18	93.3	117.3
R1.4	6	6	258.2	1 SWE	32.3	44.9	77.2	24	93.3	123.3
R2.4	6	6	105.1	1 SWE	13.1	72.8	85.9	24	93.3	129.3
R2.1	6	6	132.8	1 SWE	16.6	69.3	85.9	24	93.3	135.3
R1.5	6	6	200.7	1 SWE	25.1	61.3	86.4	30	93.3	141.3
R2.3	6	6	213.9	1 SWE	26.7	74.5	101.2	30	93.3	147.3
R2.2	6	6	225.9	1 SWE	28.2	77.2	105.4	30	93.3	153.3
R3.1	3.33	3.33	165.3	1 SWE	20.7	86.4	107.1	30	93.3	156.63
R3.3	3.33	3.33	172.2	1 SWE	21.5	85.9	107.4	30	93.3	159.96
R2.5	6	6	225.3	1 SWE	28.2	85.9	114.1	30	93.3	165.96
R3.5	3.33	3.33	175.7	1 SWE	22	107.1	129.1	30	93.3	169.29
R3.2	3.33	3.33	228.2	1 SWE	28.5	101.2	129.7	30	93.3	172.62
R3.6	3.33	3.33	188.4	1 SWE	23.6	107.4	131	30	93.3	175.95
R3.4	3.33	3.33	228	1 SWE	28.5	105.4	133.9	30	93.3	179.28

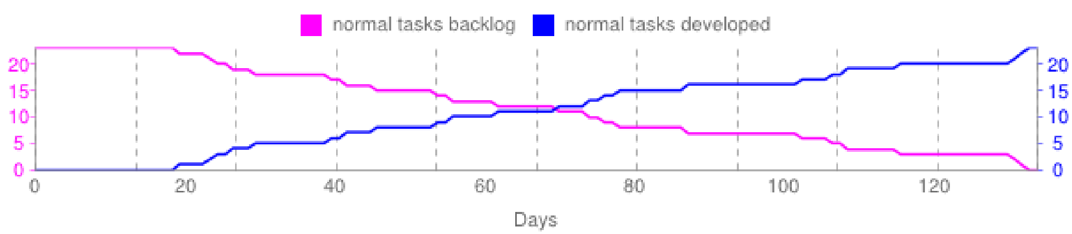


Figure 26. Product P1 Example Simulation

8.4.1 Next Steps

Next we will develop variant test cases of this example with given percent of tasks that require SE assistance. We will compare prioritization vs. no-prioritization and assess the impacts at the enterprise-level rollups.

8.4.2 Additional Utilities

Additional utilities for post-simulation Kanban board viewing have also been developed. Representative Kanban board replays are shown in Figure 27 and Figure 28, representing the process state at the timeline for a product at 35 days. These can be used step through the project events and see what happened. In the future we will allow for changing decision policies in the viewer and re-simulating.

These figures show static views of a project's Kanban boards for SE and SWE being replayed. The WIP management logic can be followed showing the policy of highest value first, when viewed in conjunction with the summary output table in Figure 5.

Some tasks have characteristics including special resource requirements. Task SE4 in Figure 27 and Task 8 in Figure 28 are specially coded flow downs of a special performance requirement requiring unique resources and tracking on the project. The task sequencing in this spin models a dependency that the SE task for this requirement be completed before the SWE task starts.

These replays of the processes are instrumented in XML output files that are used in a post viewer using JavaScript. During the distributed simulation there would be numerous Kanban boards being represented and not feasible for simultaneous interactive viewing all at once in a browser. It would be possible to select a few for interactive updates during a simulation as a potential new feature.

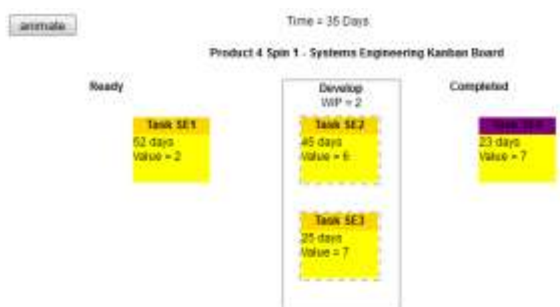


Figure 27. SE Kanban Board Animation

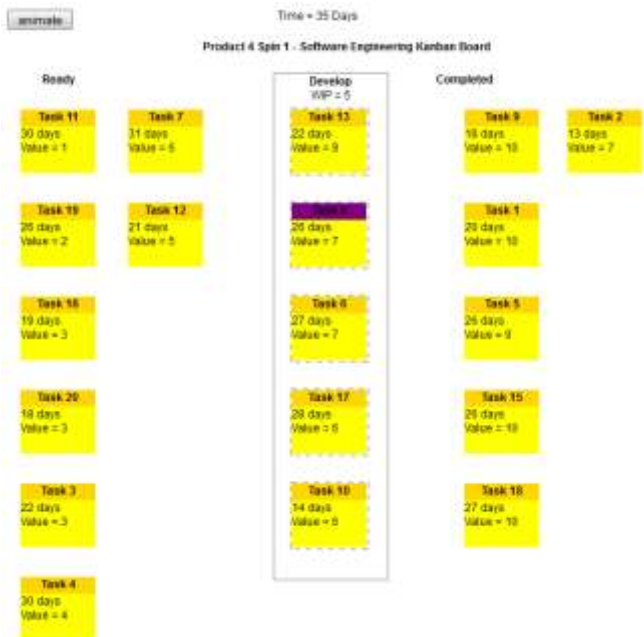


Figure 28. SWE Kanban Board Animation

9 Research Outcomes and Next Steps

9.1 Conclusions

Much of this work has been engaged in thinking through all the various scenarios that exist in highly complex system development, sustainment and evolution. We are excited about some of the concepts that have emerged and are currently developing simulations of this KSS instantiation as well as others that have occurred to us throughout the research.

One of the most difficult things to recognize while working in the systems engineering culture, is that uncertainty, while not totally unfamiliar, has not been embraced. We believe that KSS can provide more realistic understanding of work in progress, organizational capacity and can bring some statistical probability to uncertain engineering activities. The irony is that KSS designs are uncertain as well. An experience that kanban consultants and practitioners agree on is that these pull systems are rarely “engineered” and usually evolve from the first instance in ways no one expected. For that reason, we are looking forward to sowing the seeds of our ideas into the systems engineering soil and seeing the unexpected but exciting harvest that grows out of them.

9.2 Next steps for further research

Follow on work is planned to use the prototype in comparing performance with traditional SE methods. This will enable determination if SE functions are accomplished more effectively and efficiently, whether the overall value of the systems of systems over time is increased, and whether other expected results are fulfilled. Pilot projects are also planned to validate the approach in vivo.

This page intentionally left blank

10 APPENDICES

10.1 APPENDIX A: REFERENCES

1. NDIA-National Defense Industrial Association (2010). Top Systems Engineering Issues In US Defense Industry. Systems Engineering Division Task Group Report. http://www.ndia.org/Divisions/Divisions/SystemsEngineering/Documents/Studies/Top%20SE%20Issues%202010%20Report%20v11%20FINAL.pdf. September, 2010.
2. Turner, Richard, Shull F., et al (2009a) "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs: Phase 1 Final Technical Report," Systems Engineering Research Center, SERC-2009-TR002, September 2009.
3. Turner, Richard, Shull F., et al (2009b) "Evaluation of Systems Engineering Methods, Processes and Tools on Department of Defense and Intelligence Community Programs: Phase 2 Final Technical Report," Systems Engineering Research Center, SERC-2010-TR004, December 2009.
4. Turner, Richard and Wade, J. (2011). "Lean Systems Engineering within System Design Activities," Proceedings of the 3rd Lean System and Software Conference, May 2-6, 2011, Los Angeles, CA.
5. Boehm, Barry and Turner, Richard (2004). *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison Wesley.
6. Larman C. and Vodde, B. (2009). *Scaling Lean & Agile Development*. Boston, MA: Addison Wesley.
7. Poppendiek, Mary. (2007). *Implementing Lean Software Development*. Boston, MA: Addison Wesley.
8. Anderson, David. (2010). *Kanban: Successful Evolutionary Change for Your Technology Business*. Sequim, WA: Blue Hole Press
9. Reinertsen, Donald G. (2010). *The Principles of Product Development Flow*. Redondo Beach, CA: Celeritas Publishing.
10. Poppendieck, Mary, and Tom Poppendieck. (2003). *Lean Software Development: An Agile Toolkit*. The Agile Software Development Series. Boston: Addison-Wesley.
11. Morgan, James M, and Jeffrey K Liker. (2006). *The Toyota Product Development System: Integrating People, Process, and Technology*. New York: Productivity Press.

12. V. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data," IEEE Transactions on Software Engineering, vol.10(3): 728-738, November 1984.

13. Basili, Victor R. , and Carolyn Seaman, "Metric-Based Quality Management", Software Engineering for Embedded Systems Series, Fraunhofer IESE, Kaiserslautern, Germany, 2010.

14. Goldratt, Eliyahu M., and Jeff Cox. (2004). *The Goal: a Process of Ongoing Improvement*. Great Barrington, MA: North River..

15. Boehm, B. (2009). Applying the Incremental Commitment Model to Brownfield Systems Development, Proceedings, CSER 2009, April 2009.

16. Heath, B. et al. (2009). A survey of agent-based modeling practices (January 1998 to July 2008). Journal of Artificial Societies and Social Simulation. 12:4 2009.

17. Anderson et al. (2011). “Studying Lean-Kanban Approach Using Software Process Simulation.” A. Sillitti et al. (Eds.): Agile Processes in Software Engineering and Extreme Programming, Part 1, Lecture Notes in Business Information Processing, Volume 77, Pages 12-26 2011.

18. Boehm, Barry, Ricardo Valerdi, and Eric Honour (2008). “The ROI of systems engineering: Some quantitative results for software- intensive systems.” Systems Engineering 11 (3) (September 1): 221-234. doi:10.1002/sys.20096.

19. Borshchev, A., and A. Filippov (2004). "From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools." In Proceedings of the 22nd International Conference of the System Dynamics Society, 25–29.

20. M. Kellner, R. Madachy and D. Raffo (1999) *Software Process Simulation Modeling: Why? What? How?,* Journal of Systems and Software, Spring 1999.

21. R. Madachy (2008). *Software Process Dynamics*, Wiley-IEEE Press, Hoboken, NJ.

22. Office of the Deputy Under Secretary of Defense for Acquisition and Technology, Systems and Software Engineering (2008). *Systems Engineering Guide for Systems of Systems, Version 1.0*. Washington, DC: ODUSD(A&T)SSE, 2008.

23. J. Dahmann, G. Rebovich, J. Lane, R. Lowry, and K. Baldwin (2011); An Implementers' View of Systems Engineering for Systems of Systems, Proceedings of the IEEE International Systems Conference, 4-7 April, Montreal, Canada.

24. Turner, R., Madachy R., Ingold D., and Lane J., “Improving Systems Engineering Effectiveness in Rapid Response Development Environments,” Proceedings of the International Conference on Software and System Process 2012, 2012.

25. Turner, R., Madachy R., Ingold D., and Lane J., "Modeling Kanban Processes in Systems Engineering," Proceedings of the International Conference on Software and System Process 2012, 2012.

26. Turner, R., Madachy R., Ingold D., and Lane J., Anderson D., "Effectiveness of kanban approaches in systems engineering within rapid response environments," Proceedings of the Conference on Systems Engineering Research 2012, Procedia Computer Science, Vol. 8 - 2012, Elsevier, March, 2012.

27. Turner, R., "Consideration of a multi-layer on-demand scheduling system for complex, rapid-response, system-of-systems development environments," Proceedings of the 4th Lean Software and Systems Conference, May 13-16, 2012, Boston, MA; Blue Hole Press, 2012.

10.2 Appendix B – Introduction to the Kanban-based Scheduling System

10.2.1 Kanban as a starting place

A kanban (signal card) approach is a form of on-demand scheduling that provides a visual means of managing the flow within a process. The signal cards are created to the agreed capacity of the process and one card is associated with each piece of work. In manufacturing, work can mean the creation of a part, the integration of a part into an assembly, the completion of a particular analysis process, or whatever bounded and completeable activity you wish to track through the process. Once all of the cards have been associated, no more work in that process can begin until some piece of work is completed and the card becomes available. A common example of a simple kanban is the use of a limited number of tickets for entry into the Japanese Imperial Gardens [8]. The fundamental idea is to use visual signals to synchronize the flow of work with process capacity, limit the waste of work interruption, minimize excess inventory or delay due to shortage, prevent unnecessary rework, and provide a means of tracking work progress.

In knowledge work, the components of production are ideas and information [10, 11]. In software and systems, kanban systems have evolved into a means of smoothing flow by balancing work with resource capability. The concept was extended to include the limiting of work in progress according to capacity. Work cannot be started until there is an available appropriate resource. In that way, it is characterized as an on-demand or "pull" system, since the work is pulled into the activity as capacity is available rather than "pushed" via a schedule.

A kanban system is a visually monitored set of activities, where each activity has its own ready queue and set of resources to add value to work units that flow through it. The fact that queues are explicit in the system allows costs of delay and other usually invisible aspects of scheduling to be front and center in decision making. Queues also provide a vast body of experience and underlying science from the queuing theory discipline.

Control of the kanban system is generally maintained through *batch siz*e, *Work in Progress (WIP)* limits and *Classes-of-Service* (COS) definitions that prioritize work with respect to risk.

The visual representation of work is critical to kanban success, because it provides immediate understanding of the state of flow through the set of activities. This transparency makes process anomalies (both common and special cause) or resource issues easily visible, enabling the team to recognize and react immediately to resolve the issue. Flow through the kanban system is measured and tracked through statistical methods that support tuning the control parameters to improve the system. Flow measures also provide a good handle for effectiveness comparison. Because the team and management interact with the kanban board and collectively solve problems, this aspect is important in achieving continuous improvement (kaizen).

WIP is partially-completed work, equivalent to the manufacturing concept of parts inventory waiting to be processed by a production step. WIP accumulates ahead of bottlenecks unless upstream production is curtailed or the bottleneck resolved [12]. WIP in knowledge work can be roughly associated to the number of work items that have been started and not delivered. *Limiting WIP* is a concept to control flow and enhance value by specifically limiting the amount of work to be assigned to a set of resources (a WIP Limit). WIP limits accomplish several goals: they lower the context-switching overhead that impacts individuals or teams attempting to handle several simultaneous work items; they accelerate useful value by completing work in progress before starting new work; and, they provide for reasonable and sustainable resource work loads.

Using *small batch sizes* is a supporting concept to WIP. Reducing batch size limits rework and provide flexibility in scheduling and response to unforeseen change. Smaller batch sizes help stabilize the process flow and allow downstream processes to consume the batches smoothly, rather than in a start-and-stop fashion that makes inefficient use of resources. The move from “one step to glory” system initiatives to iterative, deployable increments is an example of reducing batch size. Incremental builds and ongoing, continuous integration also approximate the effect of small batch sizes.

For a different approach to describing kanban, see Mike Burrows’ *Kanban in a Nutshell* (http://positiveincline.com/index.php/2010/03/kanban-in-a-nutshell/)

In the remainder of the paper we will refer to the proposed approach as a *kanban-based scheduling system (KSS)*. While not a true kanban in the manufacturing sense, the characteristics are sufficiently similar to support the name.

10.2.2 Predicted Benefits of the Proposed Approach

A workshop was held January 27-28 2010 to discuss the development of a 3-year roadmap for transforming systems engineering. A number of issues identified and discussed in that meeting are addressed by the following benefits likely to accrue from the application of this research.

10.2.2.1 More effective integration and use of scarce systems engineering resources

Using a KSS and applying a model of SE based on continuous activities and individually requested services is a value-based way to prioritize the use of scarce SE resources across multiple projects. The value function within the next-work selection policies can be tailored to provide efficient and effective scheduling that maximizes the value provided by the resource based on multiple, system-wide parameters. Additionally, having service requests including time vs. value parameters can help determine if the delay of other service requests fulfillment is warranted by the current service request. This is addressed further under the value function discussion.

10.2.2.2 Flexibility and predictability

SE activities are generally designed for pre-specifiable, deterministic (complete and traceable) requirements and schedules. There is often an overdependence on unnecessary formal ceremony and fairly rigid schedules. Using cadence rather than schedule can provide efficient SE flow with flexibility by operating with shorter planning horizons and on-demand services. We believe that the CoS concept not only handles expedite and date-certain conditions, but also supports cross-kanban synchronization. Even though the planning is dynamic and the selection of the next piece of work to do asynchronous, we believe the use of a value-based selection function, a time-cognizant service request, customized Classes of Service, and a statistically controlled cadence provide a sufficient level of predictability where necessary.

10.2.2.3 Visibility and coordination across multiple projects

In highly concurrent engineering, the KSS provides a means of synchronizing activities across mutually dependent teams by coordinating their activities through changing value functions (work item priority) according to the degree of data completeness and maturity (risk of change). It also provides an excellent way to show where work items are and the status of work-in-progress and queued or blocked work. The ability of teams to have a common visualization of work item status also encourages a sense of collective responsibility.

In addition, the on-demand/limited planning horizon of the KSS actually reduces the impact of long latency dependency between work items by not beginning work on items that would then languish until another work item was complete.

10.2.2.4 Low governance overhead

Implementing a KSS doesn't require major changes in the way work is accomplished or imply specific organizational structures like other agile methods (e.g. Scrum). Such systems can be set up in individual projects and allowed to evolve into more effective governance over time as the project and the organization as a whole understand the best way to attain value from the practices. Even the systems engineering resource scheduling can be implemented with very little organizational impact. Practitioners make most decisions using parameters set by management (e.g. WIP limits) and their own understanding of the needs. Issues are usually identifiable from walking the visible

representation of the flow status and so are made clear to all who take part in the scheduling, including management. Metrics are inherent to the system, clearly identify problems, and track improvements. Most problems tend to be self-correcting.

10.2.2.5 Increased project and system value delivered earlier

The core rationale of most lean and agile approaches is to provide value to the customer as quickly as possible. In rapid development environments this is particularly important. By limiting WIP, more closely integrating the SE and project engineering activities, and providing both specific project and system-wide work item value determination, the KSS provides an intentional approach to achieving early value. Nevertheless, through Classes of Service, the KSS still provides for intangible or long-term investment activities to flow through the system with minimal impact on urgent activities.

10.2.3 The Kanban-based Scheduling System

In Figure 29, Figure 30, and Table 11, we define our concept of a KSS. We intend that this model be recursive at many levels to allow for complex implementations. While we currently believe work items and their associated parameters coupled with the visual representation of flow are sufficient, we may introduce new concepts to enable better communications and synchronization between the various interacting systems.

Figure 29 shows the core concept of the KSS. This core concept can be thought of as a building block or even a recursive application of the fundamentals discussed in Section 2. In general, the upstream customer for the service provided is responsible for selecting the work that enters the KSS. This is usually done collaboratively with the KSS to make sure that significant dependencies, date certain events, and other special concerns are understood. As a resource becomes available, the highest value work item is executed until it is complete, and then added to the completed work. Depending on the delivery cadence, it may go directly to the downstream consumer or it may be held until the next delivery date.

A scheduling cadence provides regular meetings of the KSS team to assess the work flow and determine if resources should be moved between activities, WIP limits adjusted, or other actions taken. Often, this is a daily activity, but the actual planning horizon selected and the nature of the work items should be used to establish the most cost effective cadence. Planning horizon is based on the visibility into upcoming work and is dependent on the WIP and ready queue limits.

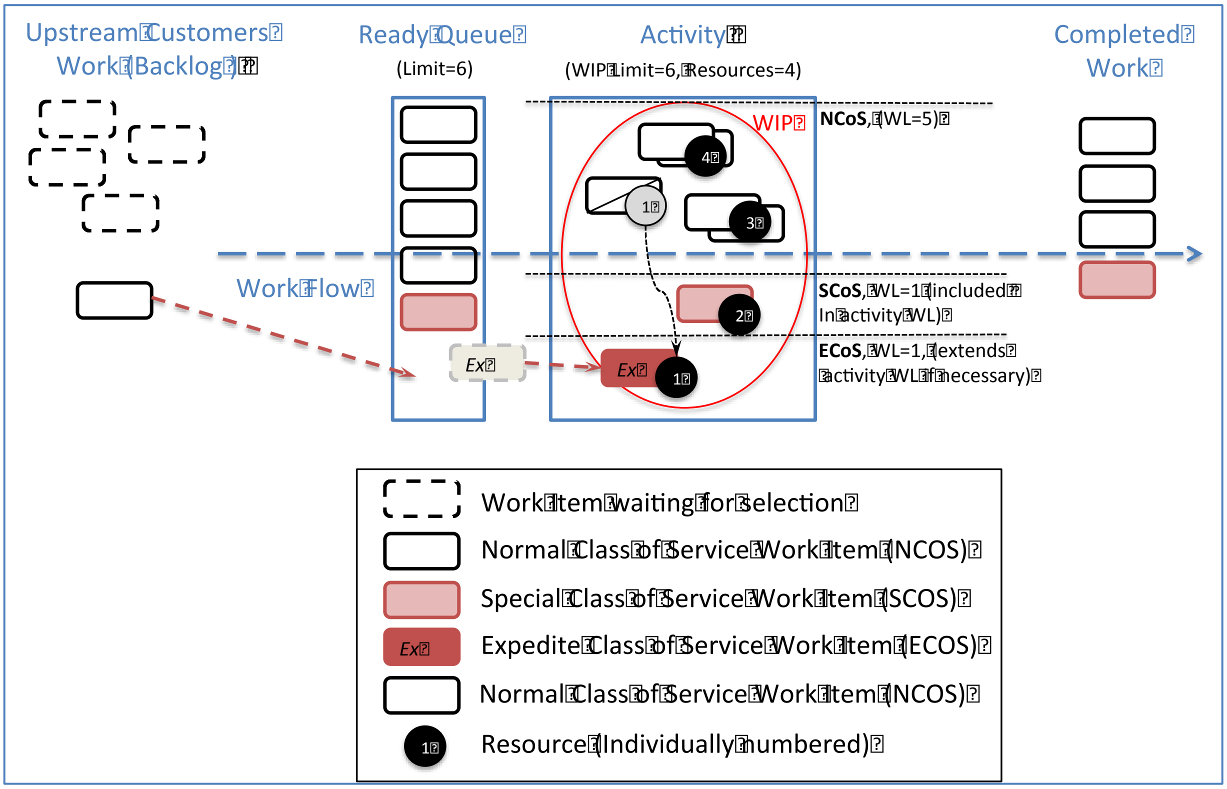


Figure 29. Kanban-based Scheduling System Model

The illustration shows a work item with a CoS of *expedite* coming into the KSS. According to the policies established for this KSS, *expedite* is allowed to bump up the WIP limit for the activity, but the activity is itself limited to only one *expedite* CoS work item at a time. The entry of the expedited work item blocks the activity from pulling any additional work items, and causes resource #1 to suspend work on the their current work item, thus blocking it as well. In this case, the team felt that resource 1 was sufficient to accomplish the expedited work item, and that allowing the remaining resources to continue their current work items best served the KSS flow. If this turned out to be wrong, adjustments could be made immediately to resolve the imbalance.

In this illustration, the KSS consists of a single activity – and that is generally how the upstream customer would view it. However, it is easy enough to see that the activity and its associated ready queue could be subdivided into multiple linked instances (see Figure 30). These could be linked sequentially or could represent different specializations for different types of services, each representing a full KSS. For example, there could be an initial activity that determines the relative value of a work item (its precedence given the current status of resources) and assigns it to the appropriate specialized service KSS.

Table 11. Kanban Scheduling System Definitions

Work Item	The item controlled in the kanban system. A work item has a definition, a Class of Service, and often a rough estimate of work effort required. The value of a work item is determined by a value function, and can vary over time (particularly for work items of special CoS such as expedite and time certain).
Effort Required	The approximate size of work in person-units of time. May be a negotiated function of desired quality.
Transit Time	The time measured from entrance of a particular work item into the KSS to its delivery to the customer.
Backlog	A customer-prioritized queue containing upstream customer work items awaiting service by a kanban system.
Cadence (prioritization and delivery)	The rhythm of the production system. Prioritization cadence defines the planning horizon for the KSS. Delivery cadence is not iteration, but allows bundling of work items for delivery. Kanban allows for iterations but decouples prioritization and delivery to allow them to vary independently of cycle time according to customer desires, domain, and costs.
Activity	Value-adding work that can be determined as complete. Includes: ready queue, a set of resources, and a WIP Limit. Allows allocation of effort to complete a work item.
Ready Queue	A limited queue that holds work items awaiting processing by an activity. The items in the queue may be considered part of the Activity WIP or the queue may have a specific limit. The queue must be bounded to maintain the kanban pull effect.
Resource	An agent for accomplishing work; may be generic or have specialized expertise. May include specific productivity. Usually associated with a specific activity, but may be shared across activities. Resources can swarm to alleviate bottlenecks or handle certain Classes of Service.
Work Selection Policies	Rules for selecting the next work item from the backlog or a ready queue when an activity has less work than its WIP limit; depends on both Class of Service and Value Function, and leads to specific flow behaviors.
Class of Service	Provides a variety of handling options for work items. May have a corresponding WIP limit for each activity to provide guaranteed access for work of that class of service. CoS WIP limit must be less than the activity's overall WIP limit. Examples are expedite, date-certain, and normal. CoS may be disruptive (e.g. expedite) and may suspend work in progress.
Value Function	Estimates the current value of a work item for use in the selection algorithm. Can be simple (null value function would produce FIFO) or a complex, multiple kanban-system, multi-factor method considering shared scarce resources and multiple cost/risk factors. Value is the means of prioritizing work items. There may be multiple value functions that return independently established values for each hierarchical layer within the KSS. For example, in SE, the overall systemic value of a work item may differ from the one that the project-level value function would return.
WIP Limit	Limit of work items allowed in progress at one time within an activity. Often initially set to twice the number of resources, but used to regulate and optimize flow and slack.
Visible Representation	A common, visual indication of workflow through the activities; often a columnar display of activities and queues. May be manual or automated. Shows status of all work-in-progress, blocked work, WIP limits. It is a characteristic that provides transparency enabling better management. Difficult to model. Provides system wide understanding of status and value, and encourages collective responsibility for flow.
Flow Metrics	Includes cumulative flow charting and average transit time.

10.2.4 Systems Engineering as a Service

Systems engineering has struggled with acceptance in rapid-response environments, partly because it tends to operate with a broader scope and with the assumption that a holistic view requires a deeper and fuller level of knowledge than is often available in the rapid response time frame. In rapid response environments, the time scale constrains the project scope, and detailed analysis up front is perceived as less achievable. Agile and lean assume holism comes from a learning process and is valuable even when incomplete.

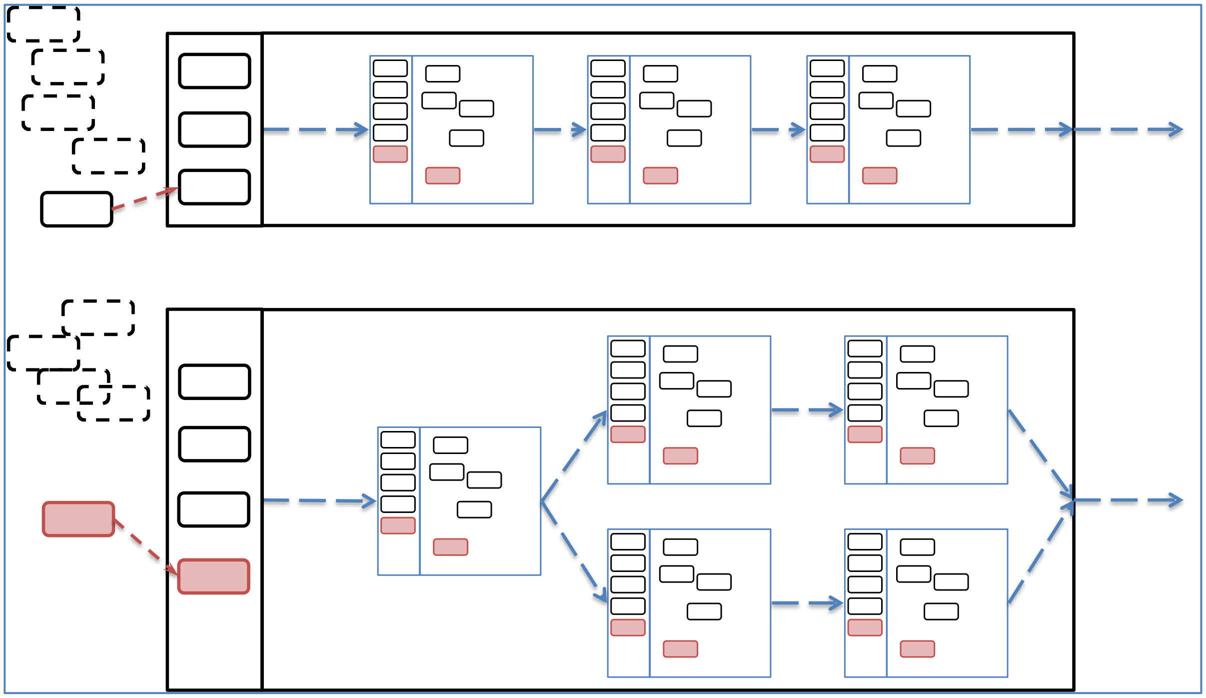


Figure 30. Kanban Scheduling System Hierarchy

The idea of using an on-demand scheduling system for systems engineering in the rapid development environment is an attempt to merge the SE flow and the software development project flow rather than simply lay SE functions on top of project activities without concern for the rapid-response constraints. Our initial model of such a system-wide KSS that includes both systems and software engineering is shown in Figure 31. We believe it will support better integration of SE into the rapid response software environment, better utilize scarce systems engineering resources, and improve the overall system-wide performance through a shared, more holistic resource allocation component.

In general, systems engineering is involved in three kinds of activities in rapid response environments: Up front, continuous, and requested. Up front activities are critical in greenfield projects, but are important in all systems and system of systems evolution. They include creating operational concepts, needs analysis, and architectural definitions. Continuous SE activities are ongoing, system–level activities (e.g. architecture, environmental risk management). These require not only substantial time, but also the maintenance and evolution of long-term, persistent artifacts that support development across multiple projects. Requested activities are generally specific to individual projects (e.g. trade studies, interface management), but will certainly draw on the persistent SE artifacts and knowledge.

By viewing the development and use of persistent artifacts as key components of services provided to various projects, SE can be opportunistic in applying its cross-project view and understanding of the larger environment to specific projects individually or in groups. It can also broker information between individual projects where there may be contractual or access barriers. When a system-wide issue or external

change occurs, SE can negotiate or unilaterally add or modify work items within affected projects to ensure that the broader issue is handled in an effective and compatible way. This is reminiscent of the agile management layer described in the iteration management approach in [13], and the approach envisioned can extend that concept throughout the rapid response lifecycle and across the multiple projects.

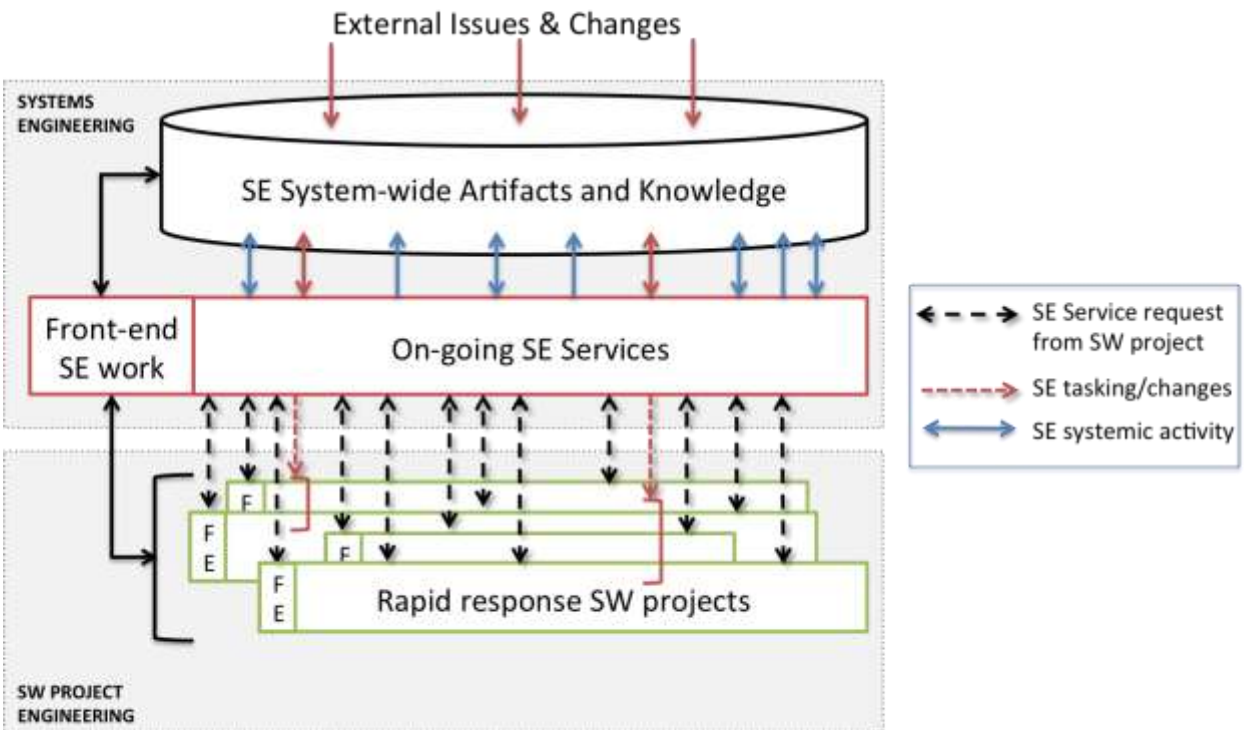


Figure 31. Overview of SE as a Service concept

SE performs its services in parallel to those activities in the requesting project and then pushes the results to the requestor as soon as available. This is aimed at supporting the timeliness of projects, so that work can continue, even if at a higher risk of rework, unless waiting for the results is blocking all other work in the project (not a good thing).

SE services require persistent artifacts and knowledge for both requestor-specific and total system artifacts/understanding. The quality of a service could be pre-specified, specified as a parameter or input with service request, or could be negotiated as a function of typical value and time available to provide the service. In a KSS, SE services can be thought of as a single activity, although some activities, particularly those up front, are likely to be complex enough to have their own set of value adding activities and specialized resources.

The value function used to select the next request to be handled must be designed to identify the highest cost of delay among the ready work items in terms of the overall system value. This allows SE to be as effective as possible in providing its services across the enterprise. The function could be based on several parameters that are attributes of individual projects, individual requests, or system-wide activities. Possibilities include

the maturity of the requesting project, lifecycle point of requesting project, criticality of the requesting project, and value/cost of delay/priority/class of service or other characteristics of the work impacted by the service requested. The details will be critical to achieve system wide benefits without impacting individual project timeliness. Only through modeling is the impact of various approaches to the value function determinable. In fact, modeling should be able to help identify the sweet spot of the amount and type of SE activity that produces the most value with the lowest impact to quality. Statistical and other measures will be needed to track the performance and improve the value function in vivo.

It should be noted, however, that developing the more general concept of SE services is outside the scope of the work documented in this report. The actual definition of services depends on the context of the projects and the development organizations. The prototype developed in this phase provides an initial description of the types of activities and the kinds of resources necessary for performing SE tasks.